Deep Learning mit Python

Lineare Regression & Gradient Decsent I

Fabian Gieseke & Moritz Seiler

Department of Information Systems
University of Münster

Übersicht

Übersicht

Notation

- Wir fassen einen Vektor $\mathbf{x} \in \mathbb{R}^d$ als Spaltenvektor auf:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

- Alternative Schreibweise: $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\mathsf{T}$

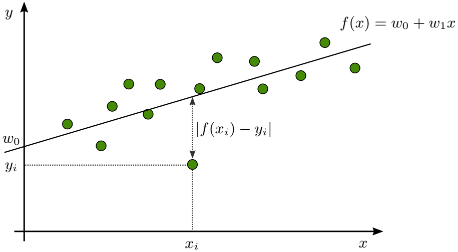
Einfache lineare Regression

- Für $d = 1$ liegen eindimensionale Merkmalsvektoren $x_i \in \mathbb{R}$ vor.

Einfache lineare Regression

- Für $d = 1$ liegen eindimensionale Merkmalsvektoren $x_i \in \mathbb{R}$ vor.
- Wir betrachten Modelle f der Form

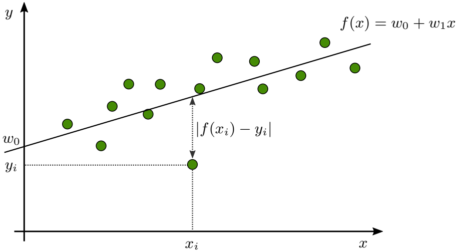
$$f(x) = f(x; w_0, w_1) = w_0 + w_1 x$$



Einfache lineare Regression

- Für $d = 1$ liegen eindimensionale Merkmalsvektoren $x_i \in \mathbb{R}$ vor.
- Wir betrachten Modelle f der Form

$$f(x) = f(x; w_0, w_1) = w_0 + w_1 x$$



- Wenn wir $\mathbf{x} := [1, x]^\mathsf{T}$ und $\mathbf{w} := [w_0, w_1]^\mathsf{T}$ festlegen, dann erhalten wir:

$$f(\mathbf{x}) = f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w}$$

Notation (Semikolon): $f(\mathbf{x}; \mathbf{w})$ bedeutet, dass unser Modell f von den Parametern $\mathbf{x}$ abhängt.

Quadratische Verlustfunktion

- Wir wollen den Fehler minimieren, den unser Modell f auf dem Trainingsdatensatz macht. Eine mögliche Fehlerfunktion ist die quadratische Verlustfunktion (*square loss*)

$$(f(x_i; w_0, w_1) - y_i)^2,$$

welche die Abweichung zwischen dem Zielwert y_i und $f(x_i; w_0, w_1)$ bestraft.

Quadratische Verlustfunktion

- Wir wollen den Fehler minimieren, den unser Modell f auf dem Trainingsdatensatz macht. Eine mögliche Fehlerfunktion ist die quadratische Verlustfunktion (*square loss*)
$$(f(x_i; w_0, w_1) - y_i)^2,$$
welche die Abweichung zwischen dem Zielwert y_i und $f(x_i; w_0, w_1)$ bestraft.
- Wir wollen einen kleinen Verlust für alle Datenpunkte erzielen, d.h.:
$$\mathcal{L}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (f(x_i; w_0, w_1) - y_i)^2$$

Quadratische Verlustfunktion

- Wir wollen den Fehler minimieren, den unser Modell f auf dem Trainingsdatensatz macht. Eine mögliche Fehlerfunktion ist die quadratische Verlustfunktion (*square loss*)

$$\left(f(x_i; w_0, w_1) - y_i\right)^2,$$

welche die Abweichung zwischen dem Zielwert y_i und $f(x_i; w_0, w_1)$ bestraft.

- Wir wollen einen kleinen Verlust für alle Datenpunkte erzielen, d.h.:

$$\mathcal{L}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i; w_0, w_1) - y_i\right)^2$$

Optimierungsziel

Finde diejenigen Parameter $\hat{w}_0$ und $\hat{w}_1$, welche den gesamten Verlust minimieren:

$$\underset{w_0, w_1}{\text{minimize}}\ \frac{1}{n}\sum_{i=1}^{n}\left(f(x_i; w_0, w_1) - y_i\right)^2$$

Quadratische Verlustfunktion

- Wir wollen den Fehler minimieren, den unser Modell f auf dem Trainingsdatensatz macht. Eine mögliche Fehlerfunktion ist die quadratische Verlustfunktion (*square loss*)

$$\left(f(x_i; w_0, w_1) - y_i\right)^2,$$

welche die Abweichung zwischen dem Zielwert y_i und $f(x_i; w_0, w_1)$ bestraft.

- Wir wollen einen kleinen Verlust für alle Datenpunkte erzielen, d.h.:

$$\mathcal{L}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i; w_0, w_1) - y_i\right)^2$$

Optimierungsziel

Finde diejenigen Parameter $\hat{w}_0$ und $\hat{w}_1$, welche den gesamten Verlust minimieren:

$$\underset{w_0, w_1}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i; w_0, w_1) - y_i\right)^2$$

Notation: $\text{minimize}_{\mathbf{x} \in S}\, g(\mathbf{x})$ beschreibt ein Optimierungsproblem mit Zielfunktion g. Ein Vektor $\mathbf{x}^*$ ist eine Lösung falls $g(\mathbf{x}^*) \leq g(\mathbf{x})$ für alle $\mathbf{x} \in S \subseteq \mathbb{R}^N$. Entsprechend: maximize

Berechnung des Modells

Optimierungsziel

Finde diejenigen Parameter $\hat{w}_0$ und $\hat{w}_1$, welche den gesamten Verlust minimieren:

$$(\hat{w}_0, \hat{w}_1) = \underset{w_0, w_1}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i; w_0, w_1) - y_i\right)^2$$

- Frage: Wie finden wir die gesuchten Modellparameter?

Berechnung des Modells

Optimierungsziel

Finde diejenigen Parameter $\hat{w}_0$ und $\hat{w}_1$, welche den gesamten Verlust minimieren:

$$(\hat{w}_0, \hat{w}_1) = \underset{w_0, w_1}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i; w_0, w_1) - y_i\right)^2$$

- Frage: Wie finden wir die gesuchten Modellparameter?
- Es liegt eine Funktion mit zwei Variablen w_0 und w_1 vor und wir suchen das Minimum bzgl. $\mathcal{L}$. Eine notwendige Bedingung ist, dass der Gradient von $\mathcal{L}$ an der Stelle $\mathbf{w} = [w_0, w_1]^\mathsf{T}$ verschwindet:

$$\nabla \mathcal{L}(w_0, w_1) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_0} \\ \frac{\partial \mathcal{L}}{\partial w_1} \end{bmatrix} \stackrel{!}{=} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Berechnung der Modellparameter

- Wir können die Zielfunktion wie folgt vereinfachen:

$$
\begin{aligned}
\mathcal{L}(w_0, w_1) &= \frac{1}{n}\sum_{i=1}^{n}\left((w_0 + x_i w_1) - y_i\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(w_0 + x_i w_1)^2 - 2(w_0 + x_i w_1)y_i + y_i^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} w_0^2 + 2w_0 x_i w_1 + x_i^2 w_1^2 - 2w_0 y_i - 2x_i w_1 y_i + y_i^2
\end{aligned}
$$

Berechnung der Modellparameter

- Wir können die Zielfunktion wie folgt vereinfachen:

$$\begin{aligned}
\mathcal{L}(w_0, w_1) &= \frac{1}{n}\sum_{i=1}^{n}((w_0 + x_i w_1) - y_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(w_0 + x_i w_1)^2 - 2(w_0 + x_i w_1)y_i + y_i^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} w_0^2 + 2w_0 x_i w_1 + x_i^2 w_1^2 - 2w_0 y_i - 2x_i w_1 y_i + y_i^2
\end{aligned}$$

- Somit erhalten wir die folgenden partiellen Ableitungen:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_0} &= 2w_0 + 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) - \frac{2}{n}\left(\sum_{i=1}^{n} y_i\right) \\
\frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) + \frac{2}{n}\left(\sum_{i=1}^{n} x_i(w_0 - y_i)\right)
\end{aligned}$$

Berechnung der Modellparameter

- $\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{n} \left(\sum_{i=1}^{n} x_i\right) - \frac{2}{n} \left(\sum_{i=1}^{n} y_i\right) \stackrel{!}{=} 0$ führt zu $\hat{w}_0 = \overline{y} - w_1 \overline{x}$.

$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$, $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, $\overline{xy} = \frac{1}{n} \sum_{i=1}^{n} x_i y_i$ und $\overline{x^2} = \frac{1}{n} \sum_{i=1}^{n} {x_i}^2$

Berechnung der Modellparameter

- $\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{n} \left(\sum_{i=1}^n x_i\right) - \frac{2}{n} \left(\sum_{i=1}^n y_i\right) \stackrel{!}{=} 0$ führt zu $\hat{w}_0 = \overline{y} - w_1 \overline{x}$.
- Einsetzen von $\hat{w}_0$ führt zu

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n} \left(\sum_{i=1}^n {x_i}^2\right) + \frac{2}{n} \left(\sum_{i=1}^n x_i(\overline{y} - w_1 \overline{x} - y_i)\right) \\
&= 2w_1 \frac{1}{n} \left(\sum_{i=1}^n {x_i}^2\right) + \overline{y} \frac{2}{n} \left(\sum_{i=1}^n x_i\right) - w_1 \overline{x} \frac{2}{n} \left(\sum_{i=1}^n x_i\right) - \frac{2}{n} \left(\sum_{i=1}^n x_i y_i\right) \\
&= 2w_1 \left(\left(\frac{1}{n} \sum_{i=1}^n {x_i}^2\right) - \overline{x}\,\overline{x}\right) + 2\overline{y}\,\overline{x} - \frac{2}{n} \left(\sum_{i=1}^n x_i y_i\right)
\end{aligned}
$$

$\overline{y} = \frac{1}{n} \sum_{i=1}^n y_i$, $\overline{x} = \frac{1}{n} \sum_{i=1}^n x_i$, $\overline{xy} = \frac{1}{n} \sum_{i=1}^n x_i y_i$ und $\overline{x^2} = \frac{1}{n} \sum_{i=1}^n {x_i}^2$

Berechnung der Modellparameter

- $\frac{\partial \mathcal{L}}{\partial w_0} = 2w_0 + 2w_1 \frac{1}{n} \left(\sum_{i=1}^n x_i\right) - \frac{2}{n} \left(\sum_{i=1}^n y_i\right) \overset{!}{=} 0$ führt zu $\hat{w}_0 = \overline{y} - w_1 \overline{x}$.
- Einsetzen von $\hat{w}_0$ führt zu

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n} \left(\sum_{i=1}^n {x_i}^2 \right) + \frac{2}{n} \left(\sum_{i=1}^n x_i (\overline{y} - w_1 \overline{x} - y_i) \right) \\
&= 2w_1 \frac{1}{n} \left(\sum_{i=1}^n {x_i}^2 \right) + \overline{y} \frac{2}{n} \left(\sum_{i=1}^n x_i \right) - w_1 \overline{x} \frac{2}{n} \left(\sum_{i=1}^n x_i \right) - \frac{2}{n} \left(\sum_{i=1}^n x_i y_i \right) \\
&= 2w_1 \left(\left(\frac{1}{n} \sum_{i=1}^n {x_i}^2 \right) - \overline{x}\,\overline{x} \right) + 2\overline{y}\,\overline{x} - \frac{2}{n} \left(\sum_{i=1}^n x_i y_i \right)
\end{aligned}
$$

- $\frac{\partial \mathcal{L}}{\partial w_1} \overset{!}{=} 0$ führt zu $\hat{w}_1 = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - (\overline{x})^2}$

$\overline{y} = \frac{1}{n} \sum_{i=1}^n y_i$, $\overline{x} = \frac{1}{n} \sum_{i=1}^n x_i$, $\overline{xy} = \frac{1}{n} \sum_{i=1}^n x_i y_i$ und $\overline{x^2} = \frac{1}{n} \sum_{i=1}^n {x_i}^2$

Berechnung der Modellparameter

- Die partiellen Ableitungen sind gegeben durch:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_0} &= 2w_0 + 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) - \frac{2}{n}\left(\sum_{i=1}^{n} y_i\right) \\ \frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) + \frac{2}{n}\left(\sum_{i=1}^{n} x_i(w_0 - y_i)\right) \end{aligned}$$

Berechnung der Modellparameter

- Die partiellen Ableitungen sind gegeben durch:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_0} &= 2w_0 + 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) - \frac{2}{n}\left(\sum_{i=1}^{n} y_i\right) \\ \frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) + \frac{2}{n}\left(\sum_{i=1}^{n} x_i(w_0 - y_i)\right) \end{aligned}$$

- Wir können auch die Hesse-Matrix at dem Punkt $(\hat{w}_0, \hat{w}_1)$ betrachten:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix} = \begin{bmatrix} 2 & \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right) \\ \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right) & \frac{2}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) \end{bmatrix}$$

Es gilt $D = 2 \cdot \left(\frac{2}{n}\sum_{i=1}^{n} {x_i}^2\right) - \left(\frac{2}{n}\sum_{i=1}^{n} x_i\right)^2 = 4 \cdot \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 > 0$ und $\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} > 0$ falls nicht alle x_i identisch sind (was wir hier annehmen).

Berechnung der Modellparameter

- Die partiellen Ableitungen sind gegeben durch:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_0} &= 2w_0 + 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) - \frac{2}{n}\left(\sum_{i=1}^{n} y_i\right)\\ \frac{\partial \mathcal{L}}{\partial w_1} &= 2w_1 \frac{1}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) + \frac{2}{n}\left(\sum_{i=1}^{n} x_i(w_0 - y_i)\right)\end{aligned}$$

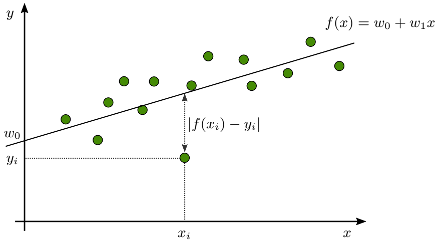
- Wir können auch die Hesse-Matrix at dem Punkt $(\hat{w}_0, \hat{w}_1)$ betrachten:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_1} \\ \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_0} & \frac{\partial^2 \mathcal{L}}{\partial w_1 \partial w_1} \end{bmatrix} = \begin{bmatrix} 2 & \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right) \\ \frac{2}{n}\left(\sum_{i=1}^{n} x_i\right) & \frac{2}{n}\left(\sum_{i=1}^{n} {x_i}^2\right) \end{bmatrix}$$

Es gilt $D = 2 \cdot \left(\frac{2}{n}\sum_{i=1}^{n} {x_i}^2\right) - \left(\frac{2}{n}\sum_{i=1}^{n} x_i\right)^2 = 4 \cdot \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 > 0$ und $\frac{\partial^2 \mathcal{L}}{\partial w_0 \partial w_0} > 0$ falls nicht alle x_i identisch sind (was wir hier annehmen).

- Somit gilt, dass an der Stelle $(\hat{w}_0, \hat{w}_1)$ ein lokales Minimum vorliegt! Man kann auch zeigen, dass dies ein globales Minimum ist ($\rightarrow$ Konvexität von Funktionen).

Univariate Lineare Regression ($d = 1$)



Lösung

Die Parameter $\hat{w}_0 = \overline{y} - \hat{w}_1 \overline{x}$ und $\hat{w}_1 = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - (\overline{x})^2}$ sind eine Lösung für:

$$\underset{w_0, w_1}{\text{minimize}} \frac{1}{n} \sum_{i=1}^{n} \left(f(x_i; w_0, w_1) - y_i \right)^2$$

Übersicht

Matrizenschreibweise

- Modell: $f(x; w_0, w_1) = f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w}$ mit $\mathbf{x} = [1, x]^\mathsf{T}$ und $\mathbf{w} = [w_0, w_1]^\mathsf{T}$

Matrizenschreibweise

- Modell: $f(x; w_0, w_1) = f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w}$ mit $\mathbf{x} = [1, x]^\mathsf{T}$ und $\mathbf{w} = [w_0, w_1]^\mathsf{T}$
- Entsprechende Erweiterung aller Datenpunkte $x_1, x_2, \ldots, x_n$ führt zu::

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix} \in \mathbb{R}^{n \times 2} \qquad \text{und} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

Matrizenschreibweise

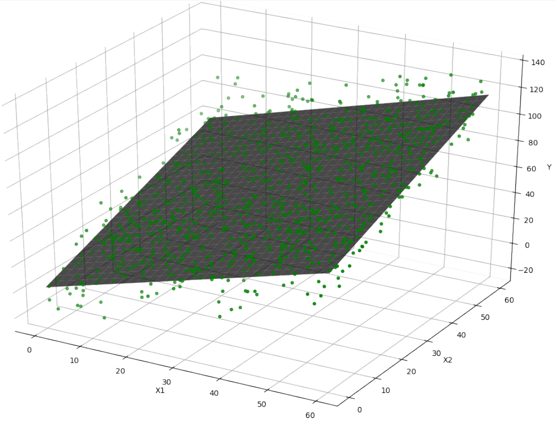
- Modell: $f(x; w_0, w_1) = f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w}$ mit $\mathbf{x} = [1, x]^\mathsf{T}$ und $\mathbf{w} = [w_0, w_1]^\mathsf{T}$
- Entsprechende Erweiterung aller Datenpunkte $x_1, x_2, \ldots, x_n$ führt zu::

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{bmatrix} \in \mathbb{R}^{n \times 2} \qquad \text{und} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

- Die Verlustfunktion kann daher geschrieben werden als:

$$\mathcal{L}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} ((w_0 + x_i w_1) - y_i)^2 = \frac{1}{n} (\mathbf{Xw} - \mathbf{y})^\mathsf{T} (\mathbf{Xw} - \mathbf{y})$$

Multiple Lineare Regression



Allgemeine (mehrdimensionale) Form

- Gegeben: Trainingsdatensatz $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$
- Ziel: Lineares Modell der Form $f(\mathbf{z}; \mathbf{w}) = w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_d z_d$

Multiple Lineare Regression

- Gegeben: Trainingsdatensatz $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$
- Erweiterung aller Datenpunkte führt zu:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,d} \\ \vdots & & & & \\ 1 & x_{n,1} & x_{n,2} & \ldots & x_{n,d} \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

Multiple Lineare Regression

- Gegeben: Trainingsdatensatz $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$
- Erweiterung aller Datenpunkte führt zu:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,d} \\ \vdots & & & & \\ 1 & x_{n,1} & x_{n,2} & \ldots & x_{n,d} \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

- Wie zuvor erhalten wir mit $\mathbf{y} = [y_1, \ldots, y_n]^\mathsf{T} \in \mathbb{R}^n$:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i; \mathbf{w}) - y_i)^2 = \frac{1}{n} (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T} (\mathbf{X}\mathbf{w} - \mathbf{y})$$

Multiple Lineare Regression

- Gegeben: Trainingsdatensatz $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$
- Erweiterung aller Datenpunkte führt zu:

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \ldots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \ldots & x_{2,d} \\ \vdots & & & & \\ 1 & x_{n,1} & x_{n,2} & \ldots & x_{n,d} \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

- Wie zuvor erhalten wir mit $\mathbf{y} = [y_1, \ldots, y_n]^\mathsf{T} \in \mathbb{R}^n$:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathbf{x}_i; \mathbf{w}) - y_i\right)^2 = \frac{1}{n} (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T} (\mathbf{X}\mathbf{w} - \mathbf{y})$$

- Ziel: Bestimmung von Modellparametern $\hat{\mathbf{w}} \in \mathbb{R}^{d+1}$, welche eine Lösung sind für:

$$\underset{\mathbf{w}}{\text{minimize}}\, \mathcal{L}(\mathbf{w})$$

Vereinfachung der Zielfunktion

$$\mathcal{L}(\mathbf{w}) \quad = \quad \frac{1}{n}(\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Vereinfachung der Zielfunktion

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \frac{1}{n}(\mathbf{X}\mathbf{w}-\mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w}-\mathbf{y}) \\ &= \frac{1}{n}((\mathbf{X}\mathbf{w})^\mathsf{T}-\mathbf{y}^\mathsf{T})(\mathbf{X}\mathbf{w}-\mathbf{y})\end{aligned}$$

Vereinfachung der Zielfunktion

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \frac{1}{n}(\mathbf{X}\mathbf{w}-\mathbf{y})^{\intercal}(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}((\mathbf{X}\mathbf{w})^{\intercal}-\mathbf{y}^{\intercal})(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}(\mathbf{X}\mathbf{w})^{\intercal}\mathbf{X}\mathbf{w}-\frac{1}{n}\mathbf{y}^{\intercal}\mathbf{X}\mathbf{w}-\frac{1}{n}(\mathbf{X}\mathbf{w})^{\intercal}\mathbf{y}+\frac{1}{n}\mathbf{y}^{\intercal}\mathbf{y}
\end{aligned}
$$

Vereinfachung der Zielfunktion

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \frac{1}{n}(\mathbf{X}\mathbf{w}-\mathbf{y})^\intercal(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}((\mathbf{X}\mathbf{w})^\intercal-\mathbf{y}^\intercal)(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}(\mathbf{X}\mathbf{w})^\intercal\mathbf{X}\mathbf{w}-\frac{1}{n}\mathbf{y}^\intercal\mathbf{X}\mathbf{w}-\frac{1}{n}(\mathbf{X}\mathbf{w})^\intercal\mathbf{y}+\frac{1}{n}\mathbf{y}^\intercal\mathbf{y} \\
&= \frac{1}{n}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w}-\frac{2}{n}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y}+\frac{1}{n}\mathbf{y}^\intercal\mathbf{y}
\end{aligned}
$$

Vereinfachung der Zielfunktion

$y^T \; X \; w - (d+1) \times 1$

$1 \times n \quad n \times (1+d)$

$$
\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \frac{1}{n}(\mathbf{X}\mathbf{w}-\mathbf{y})^\top(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}((\mathbf{X}\mathbf{w})^\top-\mathbf{y}^\top)(\mathbf{X}\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{n}(\mathbf{X}\mathbf{w})^\top\mathbf{X}\mathbf{w}-\frac{1}{n}\mathbf{y}^\top\mathbf{X}\mathbf{w}-\frac{1}{n}(\mathbf{X}\mathbf{w})^\top\mathbf{y}+\frac{1}{n}\mathbf{y}^\top\mathbf{y} \\
&= \frac{1}{n}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}-\frac{2}{n}\mathbf{w}^\top\mathbf{X}^\top\mathbf{y}+\frac{1}{n}\mathbf{y}^\top\mathbf{y}
\end{aligned}
$$

Der letzte Schritt folgt aus $\mathbf{y}^\top\mathbf{X}\mathbf{w} = ((\mathbf{X}\mathbf{w})^\top\mathbf{y})^\top \in \mathbb{R}$.

Gradient & Stationärer Punkt

Verlustfunktion

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{w}^\intercal\mathbf{X}^\intercal\mathbf{y} + \frac{1}{n}\mathbf{y}^\intercal\mathbf{y}$$

Gradient & Stationärer Punkt

Verlustfunktion

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \frac{1}{n}\mathbf{y}^\mathsf{T}\mathbf{y}$$

Toolbox (Tabelle 1.4 in Rogers & Girolami [RG16])

1. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
2. $f(\mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
3. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{w}$
4. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{C}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{C}\mathbf{w}$ (falls C symmetrisch)

Gradient & Stationärer Punkt

Verlustfunktion

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \frac{1}{n}\mathbf{y}^\mathsf{T}\mathbf{y}$$

Toolbox (Tabelle 1.4 in Rogers & Girolami [RG16])

1. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
2. $f(\mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
3. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{w}$
4. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{C}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{C}\mathbf{w}$ (falls C symmetrisch)

Aufgabe: Bestimmung des Gradienten für $\mathcal{L}(\mathbf{w})$

Gradient & Stationärer Punkt

Verlustfunktion

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \frac{1}{n}\mathbf{y}^\mathsf{T}\mathbf{y}$$

Toolbox (Tabelle 1.4 in Rogers & Girolami [RG16])

1. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{x} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
2. $f(\mathbf{w}) = \mathbf{x}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
3. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{w}$
4. $f(\mathbf{w}) = \mathbf{w}^\mathsf{T}\mathbf{C}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{C}\mathbf{w}$ (falls C symmetrisch)

Der Gradient ist $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{y}$.

Gradient & Stationärer Punkt

Verlustfunktion

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \frac{1}{n}\mathbf{y}^\top \mathbf{y}$$

Toolbox (Tabelle 1.4 in Rogers & Girolami [RG16])

1. $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{x} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
2. $f(\mathbf{w}) = \mathbf{x}^\top \mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = \mathbf{x}$
3. $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{w}$
4. $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{C}\mathbf{w} \Rightarrow \nabla f(\mathbf{w}) = 2\mathbf{C}\mathbf{w}$ (falls C symmetrisch)

(d+1)×n

n×1

(d+1)×1

Der Gradient ist $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\top \mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\top \mathbf{y}$. Deshalb:

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{w}) &= \mathbf{0} \\ \Leftrightarrow\quad & \tfrac{2}{n}\mathbf{X}^\top \mathbf{X}\mathbf{w} - \tfrac{2}{n}\mathbf{X}^\top \mathbf{y} &= \mathbf{0} \\ \Leftrightarrow\quad & \mathbf{X}^\top \mathbf{X}\mathbf{w} &= \mathbf{X}^\top \mathbf{y} \end{aligned}$$

Gradient & Stationärer Punkt

Der Gradient ist $\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{y}$. Deshalb:

$$\begin{aligned} & \nabla \mathcal{L}(\mathbf{w}) &&= \mathbf{0} \\ \Leftrightarrow \quad & \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{y} &&= \mathbf{0} \\ \Leftrightarrow \quad & \mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &&= \mathbf{X}^\mathsf{T}\mathbf{y} \end{aligned}$$

Gradient & Stationärer Punkt

Der Gradient ist $\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\top\mathbf{y}$. Deshalb:

$$\begin{aligned} & \nabla \mathcal{L}(\mathbf{w}) && = \mathbf{0} \\ \Leftrightarrow \quad & \tfrac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \tfrac{2}{n}\mathbf{X}^\top\mathbf{y} && = \mathbf{0} \\ \Leftrightarrow \quad & \mathbf{X}^\top\mathbf{X}\mathbf{w} && = \mathbf{X}^\top\mathbf{y} \end{aligned}$$

Schließlich können wir beide Seiten der Gleichung (von links) mit $(\mathbf{X}^\top\mathbf{X})^{-1}$ multiplizieren (falls $(\mathbf{X}^\top\mathbf{X})^{-1}$ existiert). Dies ergibt:

$$\mathbf{I}\mathbf{w} \quad = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

wobei $\mathbf{I} = (\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})$ die Identitätsmatrix ist.

Gradient & Stationärer Punkt

Der Gradient ist $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\top\mathbf{y}$. Deshalb:

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{w}) && = \mathbf{0} \\ \Leftrightarrow\quad & \tfrac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \tfrac{2}{n}\mathbf{X}^\top\mathbf{y} && = \mathbf{0} \\ \Leftrightarrow\quad & \mathbf{X}^\top\mathbf{X}\mathbf{w} && = \mathbf{X}^\top\mathbf{y} \end{aligned}$$

Schließlich können wir beide Seiten der Gleichung (von links) mit $(\mathbf{X}^\top\mathbf{X})^{-1}$ multiplizieren (falls $(\mathbf{X}^\top\mathbf{X})^{-1}$ existiert). Dies ergibt:

$$\mathbf{I}\mathbf{w} \quad = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

wobei $\mathbf{I} = (\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})$ die Identitätsmatrix ist. Wir erhalten somit:

Lineare Regression: Optimale Modellparameter

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Gradient & Stationärer Punkt

Der Gradient ist $\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\top\mathbf{y}$. Deshalb:

$$\begin{aligned} & \nabla\mathcal{L}(\mathbf{w}) && = \mathbf{0} \\ \Leftrightarrow\quad & \tfrac{2}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} - \tfrac{2}{n}\mathbf{X}^\top\mathbf{y} && = \mathbf{0} \\ \Leftrightarrow\quad & \mathbf{X}^\top\mathbf{X}\mathbf{w} && = \mathbf{X}^\top\mathbf{y} \end{aligned}$$

Schließlich können wir beide Seiten der Gleichung (von links) mit $(\mathbf{X}^\top\mathbf{X})^{-1}$ multiplizieren (falls $(\mathbf{X}^\top\mathbf{X})^{-1}$ existiert). Dies ergibt:

$$\mathbf{I}\mathbf{w} \quad = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

wobei $\mathbf{I} = (\mathbf{X}^\top\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{X})$ die Identitätsmatrix ist. Wir erhalten somit:

Lineare Regression: Optimale Modellparameter

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

Hinweis: Die Matrix $(\mathbf{X}^\top\mathbf{X})$ ist nicht immer invertierbar.

Vorhersagen?

Sei $\{\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_N\} \subset \mathbb{R}^d$ eine Menge von neuen Datenpunkten mit:

$$\overline{\mathbf{X}} = \begin{bmatrix} 1 & \bar{x}_{1,1} & \bar{x}_{1,2} & \dots & \bar{x}_{1,d} \\ 1 & \bar{x}_{2,1} & \bar{x}_{2,2} & \dots & \bar{x}_{2,d} \\ \vdots & & & & \\ 1 & \bar{x}_{n,1} & \bar{x}_{n,2} & \dots & \bar{x}_{N,d} \end{bmatrix} \in \mathbb{R}^{N\times(d+1)}$$

Dann erhält man die entsprechenden Vorhersagen mittels $\hat{\mathbf{y}} = \overline{\mathbf{X}}\hat{\mathbf{w}}$.

Zusammenfassung: Lineare Regression

- Gegeben: Trainingsdatensatz der Form $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$.
- Ziel: Finde $(d+1)$-dimensionalen Vektor $\hat{\mathbf{w}} = [\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_d]^\mathsf{T}$, welcher $\mathcal{L}(\mathbf{w}) = \frac{1}{n}(\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$ minimiert, d.h. welcher eine Lösung ist für:

$$\begin{aligned} \nabla\mathcal{L}(\mathbf{w}) &= \mathbf{0} \\ \Leftrightarrow \quad \mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= \mathbf{X}^\mathsf{T}\mathbf{y} \end{aligned} \tag{1}$$

Zusammenfassung: Lineare Regression

- Gegeben: Trainingsdatensatz der Form $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$.
- Ziel: Finde $(d+1)$-dimensionalen Vektor $\hat{\mathbf{w}} = [\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_d]^\mathsf{T}$, welcher $\mathcal{L}(\mathbf{w}) = \frac{1}{n}(\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$ minimiert, d.h. welcher eine Lösung ist für:

$$\begin{aligned} \nabla\mathcal{L}(\mathbf{w}) &= \mathbf{0} \\ \Leftrightarrow \quad \mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} &= \mathbf{X}^\mathsf{T}\mathbf{y} \end{aligned} \tag{1}$$

Berechnung in der Praxis (Python)

- Definition der Datenmatrix $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$
  (Numpy Arrays und Funktionen verwenden!)
- Der Gewichtsvektor $\hat{\mathbf{w}}$ kann auf verschiedene Arten bestimmt werden:
  1. Berechnung von $\hat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ (nur falls $\mathbf{X}^\mathsf{T}\mathbf{X}$ invertierbar; numerisch instabil)
  2. Lösen von (1) (z.B. mittels `numpy.linalg.solve`; nur falls $\mathbf{X}^\mathsf{T}\mathbf{X}$ invertierbar)
  3. Berechnung von $\hat{\mathbf{w}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{+}\mathbf{X}^\mathsf{T}\mathbf{y}$ (z.B. mittels `numpy.linalg.pinv`)
     $(\mathbf{X}^\mathsf{T}\mathbf{X})^{+}$ ist die *Pseudoinverse* von $\mathbf{X}^\mathsf{T}\mathbf{X}$. Falls $\mathbf{X}^\mathsf{T}\mathbf{X}$ inv., dann gilt $(\mathbf{X}^\mathsf{T}\mathbf{X})^{+} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$.
  4. Gradientenabstieg
  5. ...
- Für eine neue Menge $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N\} \subset \mathbb{R}^d$ an Datenpunkten: Berechne $\hat{\mathbf{y}} = \overline{\mathbf{X}}\hat{\mathbf{w}}$

Numerisch stabil + möglich wenn $\mathbf{X}^\mathsf{T}\mathbf{X}$ singulär (nicht invertierbar): Optionen [3] und [4]

Übersicht

Implementation in Python (Aufgabe + Pause)

jupyter California Housing - Linear Regression (Task) Last Checkpoint: 3 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
        Returns
        -------
        predictions : Array of shape [n_samples, 1]
        """

        # make sure that we have multidimensional numpy arrays
        X = numpy.array(X).reshape((X.shape[0], -1))

        # TODO: Your code here!

        return predictions
```

We first instantiate the "model" object. Afterwards, we call the "fit" method to fit our model (i.e., to compute the weights).

In [ ]:
```
model = LinearRegression()
model.fit(X, y)
```

Given the fitted model, we can now obtain predictions for new data points. For simplification, we just use our data points again here.

In [ ]:
```
preds = model.predict(X)
```

Finally, we have a look at the quality of our model by computing the RMSE and by generating a plot "predictions" vs. "true labels".

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter

from sklearn.metrics import mean_squared_error

# compute RMSE
print("RMSE: {}".format(numpy.sqrt(mean_squared_error(y, preds))))

# visualize predictions vs. true labels
fig = plt.figure(figsize=(8,8))
plt.scatter(preds, y, color="blue", alpha=0.5)
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
plt.plot([-100000,600000], [-100000, 600000], 'k--')
```

Implementation in Python (Aufgabe + Pause)

jupyter California Housing - Linear Regression (Task) Last Checkpoint: 3 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
    Returns
    -------
    predictions : Array of shape [n_samples, 1]
    """

    # make sure that we have multidimensional numpy arrays
    X = numpy.array(X).reshape((X.shape[0], -1))

    # TODO: Your code here!

    return predictions
```

We first instantiate the "model" object. Afterwards, we call the "fit" method to fit our model (i.e., to compute the weights).

Vorhersagen

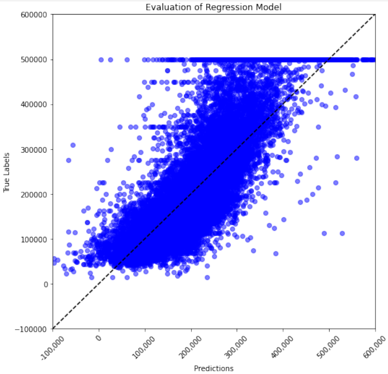
Sei $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N\} \subset \mathbb{R}^d$ eine Menge von neuen Datenpunkten mit:

$$\overline{\mathbf{X}} = \begin{bmatrix} 1 & \bar{x}_{1,1} & \bar{x}_{1,2} & \ldots & \bar{x}_{1,d} \\ 1 & \bar{x}_{2,1} & \bar{x}_{2,2} & \ldots & \bar{x}_{2,d} \\ \vdots & & & & \\ 1 & \bar{x}_{n,1} & \bar{x}_{n,2} & \ldots & \bar{x}_{N,d} \end{bmatrix} \in \mathbb{R}^{N\times(d+1)}$$

Dann erhält man die entsprechenden Vorhersagen mittels $\hat{\mathbf{y}} = \overline{\mathbf{X}}\hat{\mathbf{w}}$.

```
plt.scatter(preds, y, color="blue", alpha=0.5)
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_formatter(StrMethodFormatter('{x:,.0f}'))
plt.plot([-100000,600000], [-100000, 600000], 'k--')
```

Güte von Regressionsmodellen?



Fehlermaße

- Mean squared error (MSE): $\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i; \mathbf{w})\right)^2$
- Root mean squared error (RMSE): $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(\mathbf{x}_i; \mathbf{w})\right)^2}$
- ...

Gradientenverfahren

- Bekannt: Der Gradient

$$\nabla\mathcal{L}(\mathbf{w}) = \left[\frac{\partial\mathcal{L}}{\partial w_0}(\mathbf{w}), \ldots, \frac{\partial\mathcal{L}}{\partial w_d}(\mathbf{w})\right]^\mathsf{T}$$

  zeigt in die Richtung des steilsten Anstiegs.

Gradientenverfahren

- Bekannt: Der Gradient

$$\nabla\mathcal{L}(\mathbf{w}) = \left[\frac{\partial\mathcal{L}}{\partial w_0}(\mathbf{w}), \dots, \frac{\partial\mathcal{L}}{\partial w_d}(\mathbf{w})\right]^\mathsf{T}$$

  zeigt in die Richtung des steilsten Anstiegs.
- Idee: Wir starten mit einem initialen $\mathbf{w}$ und passen die Gewichte iterativ an:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$$

Gradientenverfahren

Bildquelle: https://scipython.com/blog/visualizing-the-gradient-descent-method/

- Bekannt: Der Gradient

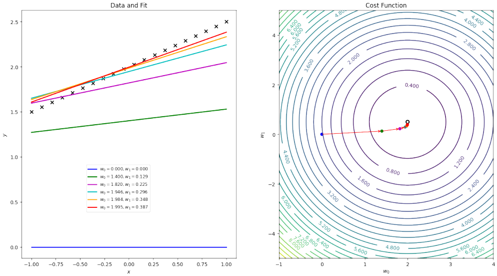
$$\nabla\mathcal{L}(\mathbf{w}) = \left[\frac{\partial\mathcal{L}}{\partial w_0}(\mathbf{w}), \dots, \frac{\partial\mathcal{L}}{\partial w_d}(\mathbf{w})\right]^\mathsf{T}$$

zeigt in die Richtung des steilsten Anstiegs.

- Idee: Wir starten mit einem initialen $\mathbf{w}$ und passen die Gewichte iterativ an:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$$

- Für $\Delta\mathbf{w}$ wird oft $\Delta\mathbf{w} = -\eta\nabla\mathcal{L}(\mathbf{w})$ verwendet (Richtung des steilsten Abstiegs), wobei $\eta > 0$ die sogenannte Lernrate ist.



Gradientenverfahren

Bildquelle: `https://scipython.com/blog/visualizing-the-gradient-descent-method/`

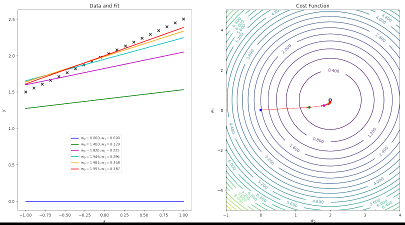
Linear Regression (batch gradient descent)

Require: Training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$ and learning rate $\eta > 0$.
Ensure: Weights $\mathbf{w}$ for the linear model $f(\mathbf{z}; \mathbf{w}) = w_0 + w_1 z_1 + w_2 z_2 + \ldots + w_d z_d$

```
// extend data matrix by prepending column of ones
// ...
// small random values (e.g., normally distributed)
Initialize w ∈ ℝ^{d+1}
repeat
    // gradient of the loss function
    Compute ∇L(w) = (2/n) XᵀXw − (2/n) Xᵀy
    // model parameter update
    w ← w − η∇L(w)
until stopping criterion is met
```

$$\nabla\mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\mathsf{T}\mathbf{y}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla\mathcal{L}(\mathbf{w})$$



Implementation in Python (Gradient Descent)

jupyter California Housing - Linear Regression (Gradient Descent, Task) Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Markdown

```python
        """

        # make sure that we have multidimensional numpy arrays
        X = numpy.array(X).reshape((X.shape[0], -1))
        # IMPORTANT: Make sure that we have a column vector!
        y = numpy.array(y).reshape((len(y), 1))

        # prepend a column of ones
        ones = numpy.ones((X.shape[0], 1))
        X = numpy.concatenate((ones, X), axis=1)

        # compute weights
        if self.algorithm == "solve":

            XtX_pinv = numpy.linalg.pinv(numpy.dot(X.T, X))
            Xty = numpy.dot(X.T,y)
            self._w = numpy.dot(XtX_pinv, Xty)

        elif self.algorithm == "gradient":

            # initialize with zeros (alternatively: small random values)
            self._w = numpy.zeros((X.shape[1], 1))

            # TODO: YOUR CODE HERE

        else:

            raise Exception("Algorithm {} not implemented!".format(self.algorithm))

    def predict(self, X, add_ones=True):
        """
        Computes predictions for a new set of points.

        Parameters
        ----------
        X : Array of shape [n_samples, n_features]

        Returns
```

(Mini-)Batch Gradient Descent

- Sei $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$ ein Trainingsdatensatz.

(Mini-)Batch Gradient Descent

- Sei $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$ ein Trainingsdatensatz.
- Batch gradient descent: Der Gradient

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{X} \mathbf{w} - \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{y}$$

basiert auf allen n Trainingsinstanzen (pro Schritt).

(Mini-)Batch Gradient Descent

- Sei $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$ ein Trainingsdatensatz.
- Batch gradient descent: Der Gradient

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{X} \mathbf{w} - \frac{2}{n} \mathbf{X}^\mathsf{T} \mathbf{y}$$

  basiert auf allen n Trainingsinstanzen (pro Schritt).
- Mini-batch gradient descent: Der Gradient basiert pro Schritt auf einer zufällig ausgewählten Teilmenge $S \subset T$ von Trainingsinstanzen.

(Mini-)Batch Gradient Descent

- Sei $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$ ein Trainingsdatensatz.
- Batch gradient descent: Der Gradient

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n} \mathbf{X}^\intercal \mathbf{X} \mathbf{w} - \frac{2}{n} \mathbf{X}^\intercal \mathbf{y}$$

  basiert auf allen n Trainingsinstanzen (pro Schritt).
- Mini-batch gradient descent: Der Gradient basiert pro Schritt auf einer zufällig ausgewählten Teilmenge $S \subset T$ von Trainingsinstanzen.
  - Die Anzahl $|S|$ der Elemente wird batch size genannt.
    (Wird i.A. auf einen Wert festgelegt (z.B. $|S| = 32$))

(Mini-)Batch Gradient Descent

- Sei $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathcal{Y}$ ein Trainingsdatensatz.
- Batch gradient descent: Der Gradient

$$\nabla \mathcal{L}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\intercal \mathbf{X}\mathbf{w} - \frac{2}{n}\mathbf{X}^\intercal \mathbf{y}$$

  basiert auf allen n Trainingsinstanzen (pro Schritt).
- Mini-batch gradient descent: Der Gradient basiert pro Schritt auf einer zufällig ausgewählten Teilmenge $S \subset T$ von Trainingsinstanzen.
  - Die Anzahl $|S|$ der Elemente wird batch size genannt. (Wird i.A. auf einen Wert festgelegt (z.B. $|S| = 32$))
  - Für $|S| = 1$ erhält man die stochastic gradient descent (SGD)-Variante.

Übersicht

Zusammenfassung

Zusammenfassung

- Einfache lineare Regression, multiple lineare Regression
- Gradientenverfahren
- Implementationen in Python

Literatur I

[RG16] Simon Rogers and Mark Girolami.
A First Course in Machine Learning, Second Edition.
Chapman & Hall/CRC, 2nd edition, 2016.