*** Applied Machine Learning Fundamentals *** Regression

Daniel Wehner

SAP SE

August 13, 2019

Agenda August 13, 2019

Section:

Introduction to Regression

Regression

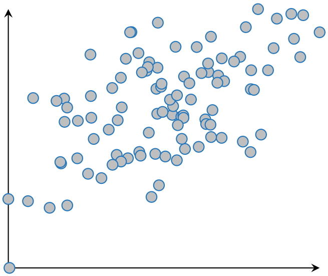
Type of target variable	Continuous
Type of training information	Supervised
Example Availability	Batch learning

Algorithm sketch: Given the training data $\mathcal{D}$ the algorithm derives a function of the type

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_{m+1} x_m \qquad \boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{\theta} \in \mathbb{R}^{m+1} \tag{1}$$

from the data. $\boldsymbol{\theta}$ is the parameter vector containing the coefficients to be estimated by the regression algorithm. Once $\boldsymbol{\theta}$ is learned it can be used for prediction.

Example Data Set: Revenues



- Find a linear function:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_{m+1} x_m$$

- Usually: $x_0 = 1$:

$$\widehat{\boldsymbol{x}} \in \mathbb{R}^{m+1} = [1 \ \boldsymbol{x}]^\intercal$$

$$h_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}) = \sum_{j=0}^{m+1} \theta_j x_j = \boldsymbol{\theta}^\intercal \widehat{\boldsymbol{x}}$$

Error Function for Regression

- In order to know how good the function fits we need an error function $\mathcal{J}(\boldsymbol{\theta})$:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} (h_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}^{(i)}) - y^{(i)})^2 \tag{2}$$

- We want to minimize $\mathcal{J}(\boldsymbol{\theta})$:

$$\min_{\boldsymbol{\theta}} \frac{1}{2n} \sum_{i=1}^{n} (h_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}^{(i)}) - y^{(i)})^2$$

- This is **ordinary least squares (OLS)**

Error Function Intuition

Section:

Solutions to Regression

Closed-Form Solutions

- Usual approach (for two unknowns): Calculate θ_0 and θ_1 according to

sample mean $\overline{x}$

$$\theta_0 = \overline{y} - \theta_2\overline{x} \qquad\qquad \theta_1 = \frac{\sum_{i=1}^{n}(x^{(i)} - \overline{x}) \cdot (y^{(i)} - \overline{y})}{\sum_{i=1}^{n}(x^{(i)} - \overline{x})^2} \tag{3}$$

- **'Normal equation'** (scales to arbitrary dimensions):

$$\boldsymbol{\theta} = \underbrace{(\widehat{\boldsymbol{X}}^\intercal \widehat{\boldsymbol{X}})^{-1} \widehat{\boldsymbol{X}}^\intercal}_{\text{Moore-Penrose pseudo-inverse}} \boldsymbol{y} \tag{4}$$

$\widehat{\boldsymbol{X}}$ is called 'design matrix' or 'regressor matrix'

Design Matrix / Regressor Matrix

- The design matrix $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times (m+1)}$ looks as follows:

$$\hat{\boldsymbol{X}} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_m^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_m^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_m^{(n)} \end{pmatrix} \tag{5}$$

In the following

$\hat{\boldsymbol{X}} \equiv \boldsymbol{X}$

- And the $n \times 1$ label vector:

$$\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, y^{(3)}, \dots, y^{(n)}\right)^{\mathsf{T}}$$

Derivation of the Normal Equation

- The derivation involves a bit of linear algebra
- Step ➊: Rewrite $\mathcal{J}(\boldsymbol{\theta})$ in matrix-vector notation:

$$
\begin{aligned}
\mathcal{J}(\boldsymbol{\theta}) &= \frac{1}{2}(\boldsymbol{X\theta} - \boldsymbol{y})^{\intercal}(\boldsymbol{X\theta} - \boldsymbol{y}) \\
&= ((\boldsymbol{X\theta})^{\intercal} - \boldsymbol{y}^{\intercal})(\boldsymbol{X\theta} - \boldsymbol{y}) \\
&= (\boldsymbol{X\theta})^{\intercal}\boldsymbol{X\theta} - (\boldsymbol{X\theta})^{\intercal}\boldsymbol{y} - \boldsymbol{y}^{\intercal}(\boldsymbol{X\theta}) + \boldsymbol{y}^{\intercal}\boldsymbol{y} \\
&= \boldsymbol{\theta}^{\intercal}\boldsymbol{X}^{\intercal}\boldsymbol{X\theta} - 2(\boldsymbol{X\theta})^{\intercal}\boldsymbol{y} + \boldsymbol{y}^{\intercal}\boldsymbol{y}
\end{aligned}
$$

- To be continued...

Derivation of the Normal Equation (Ctd.)

- Step ❷: Calculate the derivative of $\mathcal{J}(\boldsymbol{\theta})$ and set it to zero:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = 2\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^\intercal\boldsymbol{y} \stackrel{!}{=} \mathbf{0}$$
$$\Leftrightarrow \boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}^\intercal\boldsymbol{y}$$

- If $\boldsymbol{X}^\intercal\boldsymbol{X}$ is invertible, we can multiply both sides by $(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}$:

Normal equation:

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y}$$

Problems with Matrix Inversion?

- What if $(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$ does not exist?
- Problems and solutions:
  1. Linearly dependent (redundant) features or design matrix does not have full rank? (E. g. size in m^2 and size in feet^2)
     $\Rightarrow$ **Delete correlated features**
  2. Too many features ($m > n$)?
     $\Rightarrow$ **Delete features (e. g. using PCA) / add training examples**
  3. Other numerical instabilities?
     $\Rightarrow$ **Add a regularization term** (later)
  4. Computationally too expensive?
     $\Rightarrow$ **Use gradient descent**

Gradient Descent

- We want to minimize a smooth function $\mathcal{J} : \mathbb{R}^{m+1} \to \mathbb{R}$:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \mathcal{J}(\boldsymbol{\theta})$$

- Update the parameters iteratively:

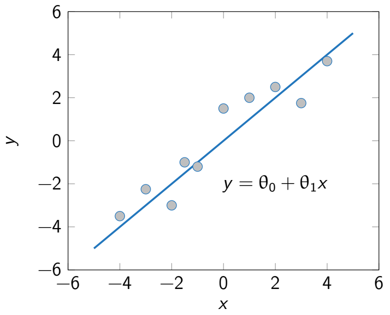
$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^{(t)}) \tag{6}$$

- where $\alpha > 0$ (learning rate) and $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ is the gradient of $\mathcal{J}(\boldsymbol{\theta})$ w. r. t. $\boldsymbol{\theta}$:

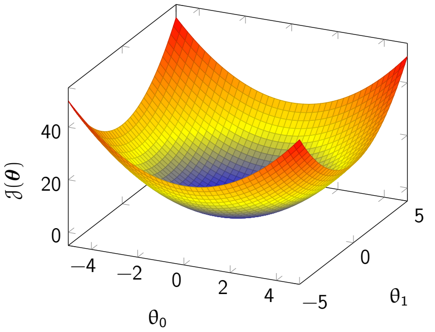
$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \left(\frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_0}, \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_1}, \dots, \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_{m+1}} \right)^{\mathsf{T}}$$

Data Input Space vs. Hypothesis Space

Data input space



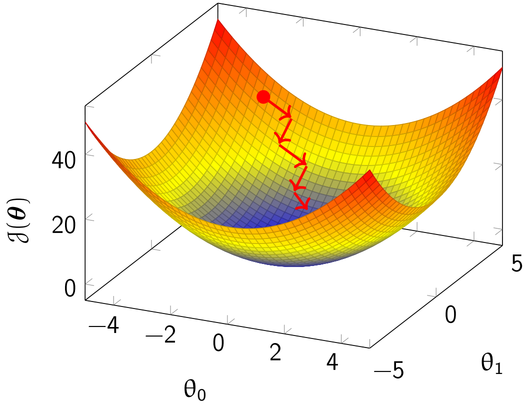
Hypothesis space $\mathcal{H}$



Data Input Space vs. Hypothesis Space (Ctd.)

- **Data input space**
  - Determined by the m **attributes** of the data set $x_1, x_2, \ldots, x_m$
  - Often high-dimensional
- **Hypothesis space** $\mathcal{H}$
  - Determined by the **number of parameters** of the model
  - Each point in the hypothesis space corresponds to a **specific assignment of model parameters**
  - The error function gives information about how good this assignment is
  - **Gradient descent is applied in the hypothesis space** $\mathcal{H}$

Visualization of Gradient Descent in 3 Dimensions



Versions of Gradient Descent

- Assume some training data $\mathcal{D}$: $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{n}$
- Squared error for a **single** example: $\ell(y_{pred}, y_{true}) = (y_{pred} - y_{true})^2$
- Our objective is to minimize the **total** error:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \sum_{i=1}^{n} \ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)})$$

- Three versions of gradient descent:
  1. Batch gradient descent
  2. Stochastic gradient descent
  3. Mini-batch gradient descent

Versions of Gradient Descent (Ctd.)

- **Batch gradient descent**: Compute gradient based on <u>**ALL**</u> data points

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \nabla \ell(h_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{7}$$

- **Stochastic gradient descent**: Compute gradient based on a <u>**SINGLE**</u> data point **(pick training example randomly and not sequentially!)**
- `For` $i \in \{1, \ldots, n\}$ `do:`

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla \ell(h_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{8}$$

Solving linear Regression using Gradient Descent

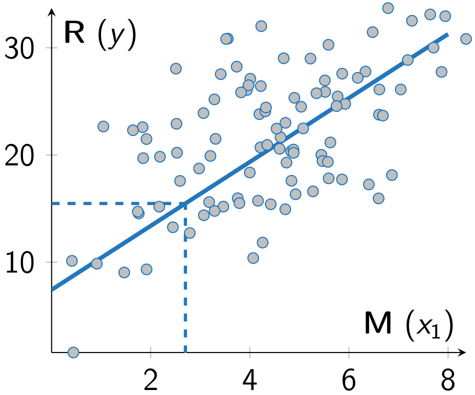
- Randomly initialize $\boldsymbol{\theta}$
- To minimize the error, keep changing $\boldsymbol{\theta}$ according to:

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^{(t)}) \tag{9}$$

- We need to calculate $\nabla_{\theta_j} \mathcal{J}(\boldsymbol{\theta}^{(t)})$: (based on a single example)

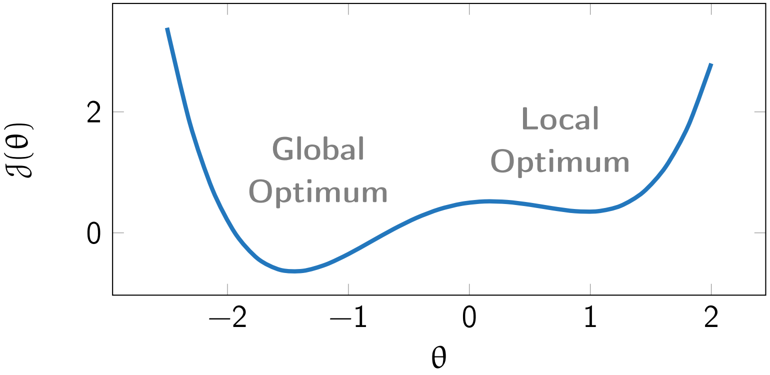
$$\begin{aligned}\nabla_{\theta_j} \mathcal{J}(\boldsymbol{\theta}^{(t)}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2 = 2 \cdot \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \\ &= (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \cdot \frac{\partial}{\partial \theta_j} (\theta_0 x_0 + \cdots + \theta_{m+1} x_{m+1} - y) = \boxed{(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) x_j}\end{aligned}$$

Solving the introductory Example



- $\theta_0 \approx 7.4218$
- $\theta_1 \approx 2.9827$
- $\mathcal{J}(\boldsymbol{\theta}) \approx 446.9584$
- $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 7.4218 + 2.9827 \cdot x_1$
- $R = h_{\boldsymbol{\theta}}(2.7) = \underline{\underline{\mathbf{15.4750}}}$

Disadvantage of Gradient Descent



Section:

Probabilistic Regression

Section:

Basis Function Regression

Summary

Lecture Overview

Unit I: Machine Learning Introduction

Self-Test Questions

Recommended Literature and further Reading

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Regression
Date: August 13, 2019

Contact:
Daniel Wehner (D062271)
SAP SE
daniel.wehner@sap.com

Do you have any questions?