Escuela Politécnica de Ingeniería de Gijón — Universidad de Oviedo
Grado en Ciencia e Ingeniería de Datos

AA1 Aprendizaje Automático 1

T4: Linear regression with gradient descent

Antonio Bahamonde
Departamento de Informática

Contents

- Linear regression
- Gradient descent
- Linear regression with gradient descent

Linear regression

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θi's are the **parameters** (also called **weights**) parameterizing the space of linear functions mappings

$$h : \boldsymbol{x} \mapsto \boldsymbol{y}$$

Linear regression

When there is no risk of confusion, we will drop the θ subscript in

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

write it more simply as h(x). To simplify our notation, we also introduce the convention of letting $x_0 = 1$ (this is the **intercept** term), so that

$$h(x) = \sum_{i=0}^{d} \theta_i x_i = \theta^T x,$$

Linear regression

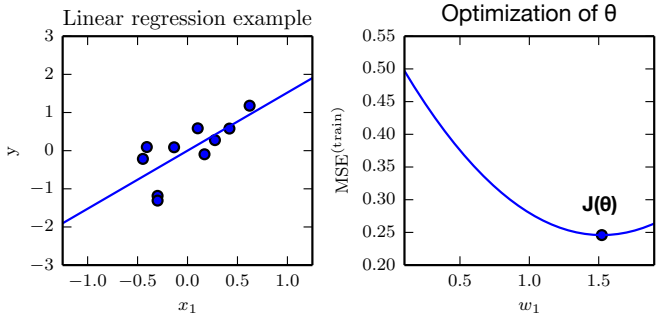
Given a training set, how do we pick, or learn, the parameters θ?
One reasonable method seems to be to make h(x) close to y, at least for the training examples we have.

To formalize this, we will define a function that measures, for each value of the θ's, how close the h(x)'s are to the corresponding y's . We define the **cost function**:

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

least-squares cost function that gives rise to the ordinary least squares regression model

Linear Regression



$y \approx \theta_1 x_1$

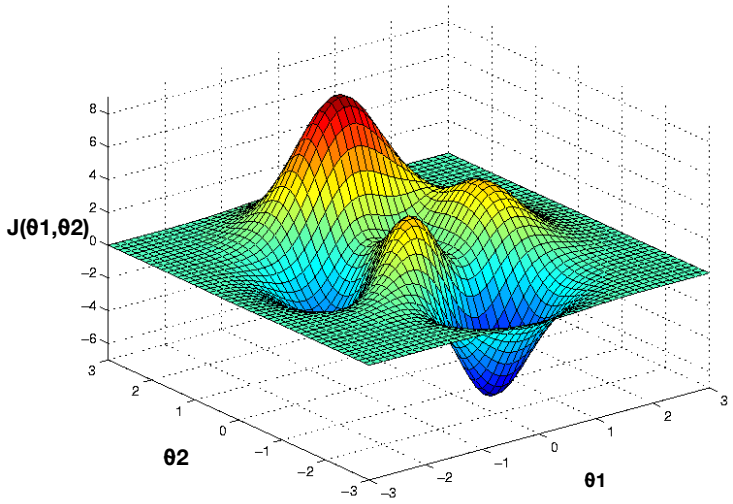
Gradient descent

Gradient descent

Given a function J(θ1,θ2) we look for the values of θ1 and θ2 that make the value of J to a minimum

Algorithm:

- Assign arbitrary values to θ1 and θ2
- Modify the values of θ1 and θ2 to reduce the value of J
  Until we believe that we have reached a minimum



Gradient Descent

To solve

$$\theta \leftarrow \underset{\theta}{\operatorname{argmin}}\ \mathbf{J}(\theta)$$

GD is the algorithm

$\theta \longleftarrow \theta_0$
repeat
$\theta \longleftarrow \theta - \gamma * \frac{\partial J}{\partial \theta}$
until no more improvement can be reached
return θ

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

Simultaneously in all the components of θ

α learning rate

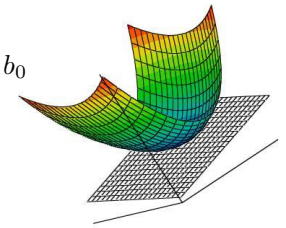
Gradient descent

The Taylor polynomial of order 1 states that for values **f(z)** when **z** is close to **a** can be approximated using the derivative as follows

$$f(z) = f(a) + (z-a)^T \nabla_z f(a)$$

That is to say,

$$f(z) = f(a) + z_1 b_1 + \ldots + z_m b_m + b_0$$
$$(b_1, \ldots, b_m) = \nabla_z f(a)$$
$$b_0 = -a^T \nabla_z f(a)$$



Gradient descent

$$f(z) = f(a) + (z-a)^T \nabla_z f(a)$$

If we state that **z** is **a** modified by a small increase (γ) in the address of a unit vector **u** ($\| \mathbf{u} \| = 1$)

$$z = a + \gamma u$$

We have that

$$f(a + \gamma u) = f(a) + \gamma u^T \nabla_z f(a)$$

Gradient descent

To find a value **z** such that **f(z) < f(a)**, we must find a direction (**u**) that

$$\begin{aligned} &\underset{u}{\operatorname{argmin}}\ \boldsymbol{u}^T \nabla_z f(\boldsymbol{a}) \\ &\quad = \underset{u}{\operatorname{argmin}}\ \|\boldsymbol{u}\| \|\nabla_z f(\boldsymbol{a})\| \cos(\boldsymbol{u}, \nabla_z f(\boldsymbol{a})) \\ &\quad = \underset{u}{\operatorname{argmin}}\ \cos(\boldsymbol{u}, |\nabla_z f(\boldsymbol{a})) \end{aligned}$$

Therefore **u** must be parallel to the oposite of the gradient (**cos = -1**)

$$\boldsymbol{u} \| -\nabla_z f(\boldsymbol{a})$$

Gradient descent

Therefore (if γ is *small*)

$$\begin{aligned} f(\boldsymbol{a} - \gamma \nabla_z f(\boldsymbol{a})) &= f(\boldsymbol{a}) - \gamma \nabla_z f(\boldsymbol{a})^T \nabla_z f(\boldsymbol{a}) \\ &= f(\boldsymbol{a}) - \gamma \|\nabla_z f(\boldsymbol{a})\|^2 \\ &< f(\boldsymbol{a}) \end{aligned}$$

After the assignment

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - \gamma \nabla_z f(\boldsymbol{a})$$

We have a point with lower f-value than f(a), whenever γ is small

Gradient descent

Tying up loose ends. The goal was to solve

$$\theta \leftarrow \underset{\theta}{\operatorname{argmin}}\ \frac{1}{|D|} \sum_{(x,y) \in D} \operatorname{loss}(y, h_\theta(\boldsymbol{x}))$$

We look for the θ value that minimizes the mean of the loss function over the dataset D

$$\begin{aligned} J(\theta) &= \frac{1}{|D|} \sum_{(x,y) \in D} \operatorname{loss}(y, h_\theta(\boldsymbol{x})) \\ &= \mathbb{E}_{(x,y) \sim p(D)} \operatorname{loss}(y, h_\theta(\boldsymbol{x})) \end{aligned}$$

Gradient descent

To solve

$$\theta \leftarrow \underset{\theta}{\operatorname{argmin}}\ \mathbf{J}(\theta)$$

GD is the algorithm

$\theta \longleftarrow \theta_0$

repeat

$\theta \longleftarrow \theta - \gamma * \frac{\partial J}{\partial \theta}$

until no more improvement can be reached

return θ

Mini Batch and SGD

GD has the problem to compute

$$\frac{\partial \mathbf{J}(\theta)}{\partial \theta} = \frac{1}{|D|} \sum_{(x,y)\in D} \frac{\partial}{\partial \theta} \text{loss}(\boldsymbol{y}, f_\theta(\boldsymbol{x}))$$

Thus, to make GD useful, we estimate this average with the mean of a subset D' of D

$$\frac{\partial \mathbf{J}(\theta)}{\partial \theta} \cong \frac{1}{|D'|} \sum_{(x,y)\in D'} \frac{\partial}{\partial \theta} \text{loss}(\boldsymbol{y}, f_\theta(\boldsymbol{x}))$$

This is called the Stochastic Gradient Descent with mini batch. Whenever D' has only one element, we have SGD

Optimization results

Note that, while gradient descent can be susceptible to **local minima in general**, the optimization problem we have posed here

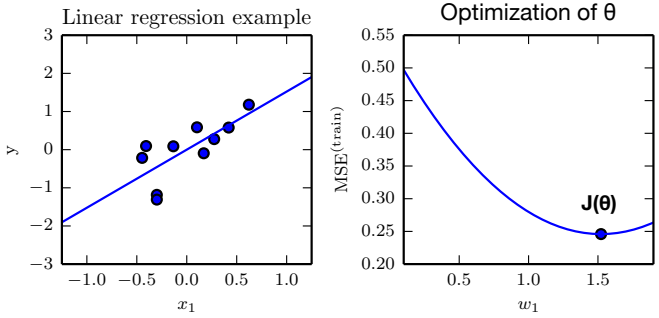
for linear regression

- has only one global, and no other local, optima; gradient descent always converges (assuming the learning rate (γ or α) is not too large) to the global minimum.
- J is a convex quadratic function.
- Stochastic gradient descent gets θ "close" to the minimum much faster than batch gradient descent.

(Note however that it may never "converge" to the minimum, and the parameters θ will keep oscillating around the minimum of J(θ); but in practice most of the values near the minimum will be reasonably good approximations to the true minimum.

Linear regression algorithm

Linear Regression



$y \approx \theta_1 x_1$

Gradient descent

To solve

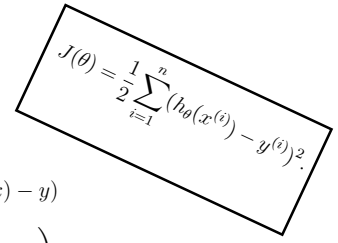
$$\theta \leftarrow \underset{\theta}{\operatorname{argmin}}\ \mathbf{J}(\theta)$$

GD is the algorithm

$\theta \longleftarrow \theta_0$
repeat
$\theta \longleftarrow \theta - \gamma * \frac{\partial J}{\partial \theta}$
until no more improvement can be reached
return θ

Linear regression

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$



$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y)\, x_j
\end{aligned}$$

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}.$$

Linear regression

For a single training example, this gives the LMS (“least mean squares”) update rule

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}.$$

the magnitude of the update is proportional to the error term ($y^{(i)} - h_\theta(x^{(i)})$):

- if we are encountering a training example on which our prediction nearly matches the actual value of $y^{(i)}$, then we find that there is little need to change the parameters;
- a larger change to the parameters will be made if our prediction has a large error

Probabilistic interpretation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}, \qquad p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right).$$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)};\theta) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Probabilistic interpretation

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2. \end{aligned}$$

Probabilistic interpretation

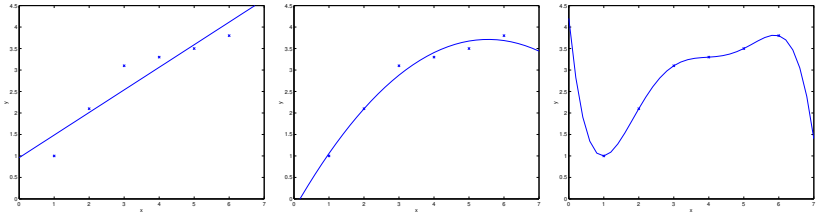
Least-squares regression corresponds to finding the maximum likelihood estimate of θ.

θ did not depend on what was σ, even if σwere unknown.

.

$$\frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2,$$

Overfitting underfitting



Más información

- Linear regression (Main notes, chapter 1)
- Linear Algebra Review and Reference. Zico Kolter (updated by Chuong Do and Tengyu Ma) June 20, 2020

  https://cs229.stanford.edu/lectures-spring2022/cs229-linear_algebra_review.pdf