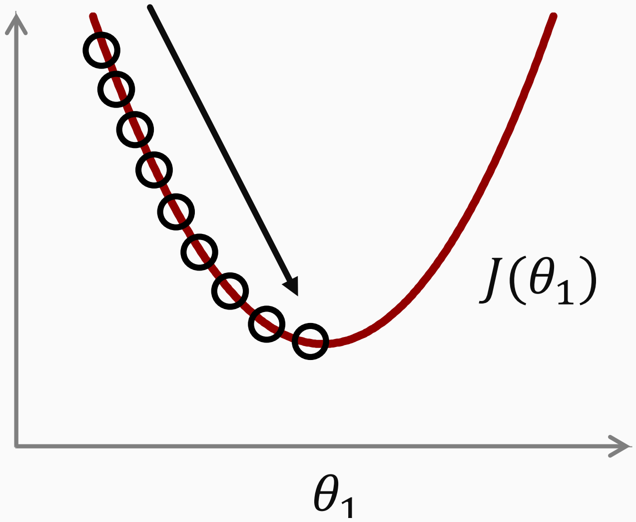
computing parameters ▪ analytically

normal equation

previously, gradient descent was applied in a fashion to take iterative steps in the direction of the global minimum of a training data set by minimizing cost function $J(\theta)$ (illustrated to the right)

alternatively, the normal equation offers a measure to solve for θ analytically without taking steps through a reiterative process algorithm



intuition if 1D ($\theta \in \mathbb{R}$):

$$J(\theta) = a\theta^2 + b\theta + c$$

they way to minimize cost functions in calculus is to take derivatives and set to 0

$\frac{\partial}{\partial\theta_1} J(\theta) = \cdots \text{set} = 0$ and solve for θ in the case where θ is a real number $\theta \in \mathbb{R}$

in the case where theta θ is a vector $n+1$;

$$\theta \in \mathbb{R}^{n+1} \qquad J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$\frac{\partial}{\partial\theta_1} J(\theta) = \cdots \text{set} = 0 \qquad \text{(for every } j) \qquad \rightarrow \qquad \text{solve for } \theta_0, \theta_1, \dots, \theta_n$$

a running example for implementation guidance of the normal equation

training examples: $m = 4$

x_0	size (feet^2) x_1	number bedrooms x_2	number floors x_3	home age (years) x_4	price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

the initial data will all be converted into a matrix with the predicted values as a vector; then setting theta θ to X transposed X inverse times X transposed y will compute the value of theta θ that minimizes the cost function

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix} \qquad \theta = (X^T X)^{-1} X^T y$$

$$\mathbb{R}^{m \times (n+1)} \qquad \mathbb{R}^m$$

in more general terms:

with $\boldsymbol{m}$ **examples** $(x^1, y^1), \ldots, (x^m, y^m)$; and $\boldsymbol{n}$ **features**

each training example x^i will represent a 1-dimensional vector:

$$x^i = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \text{ with matrix } X \text{ designed as: } X = \begin{bmatrix} -- & \left(x^{(1)}\right)^T & -- \\ -- & \left(x^{(2)}\right)^T & -- \\ -- & \left(x^{(3)}\right)^T & -- \\ -- & \vdots & -- \\ -- & \left(x^{(m)}\right)^T & -- \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}$$

e.g. if $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix} \rightarrow X = \begin{bmatrix} 1 & x_1^{(i)} \\ 1 & x_2^{(i)} \\ 1 & \vdots \\ 1 & x_m^{(i)} \end{bmatrix}$ and $y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$; then solve $\theta = (X^T X)^{-1} X^T y$

Suppose you have the training in the table below:

age (x_1)	height in cm (x_2)	weight in kg (y)
4	89	16
9	124	28
5	103	20

You would like to predict a child's weight as a function of his age and height with the model

$\text{weight} = \theta_0 + \theta_1 \text{age} + \theta_2 \text{height}$.

What are X and y?

- $X = \begin{bmatrix} 4 & 89 \\ 9 & 124 \\ 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$
- $X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 1 & 16 \\ 1 & 28 \\ 1 & 20 \end{bmatrix}$
- $X = \begin{bmatrix} 4 & 89 & 1 \\ 9 & 124 & 1 \\ 5 & 103 & 1 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$
- $X = \begin{bmatrix} 1 & 4 & 89 \\ 1 & 9 & 124 \\ 1 & 5 & 103 \end{bmatrix}, y = \begin{bmatrix} 16 \\ 28 \\ 20 \end{bmatrix}$

Correct Response

to reiterate the intuition behind the normal equation: $\theta = (X^T X)^{-1} X^T y$

$(X^T X)^{-1}$ is simply the inverse of matrix $X^T X$

set $A = X^T X$ then $A^{-1} = (X^T X)^{-1}$

in **octave** programming language, the above is denoted as: `pinv(X'*X)*X'*y`

feature scaling is **not** necessary if the normal equation method is used. however, gradient descent still has value in feature scaling.

differentiation between gradient descent and the normal equations when a dataset is with $\boldsymbol{m}$ **examples** (x^1, y^1),..., (x^m, y^m); and $\boldsymbol{n}$ **features**:

gradient descent	**normal equation**
need to choose learning rate α	no need to choose learning rate α
requires multiple iterations	requires no multiple iterations
	need to compute $(X^T X)^{-1}$
works well when n is large	works slowly when n is very large (>~10,000)

in summary, so long as the number of features $=< 1{,}000$ or so, the normal equation method is appropriate to use as opposed to a full scale gradient descent.

normal equation noninvertibility

normal equation $\theta = (X^T X)^{-1} X^T y$

what if $X^T X$ is non-invertible? (singular/degenerate)

this is a rare occurrence, however, `pinv(X'*X)*X'*y` in octave will still solve for θ

causes of $X^T X$ being non-invertible are often:

redundant features (linearly dependent)

e.g. $x_1 = \text{size in feet}^2$

$x_2 = \text{size in meters}^2$

too many features (e.g. $m \leq n$)

either delete insignificant features or use regularization