Linear Regression

Dan Sheldon

November 18, 2014

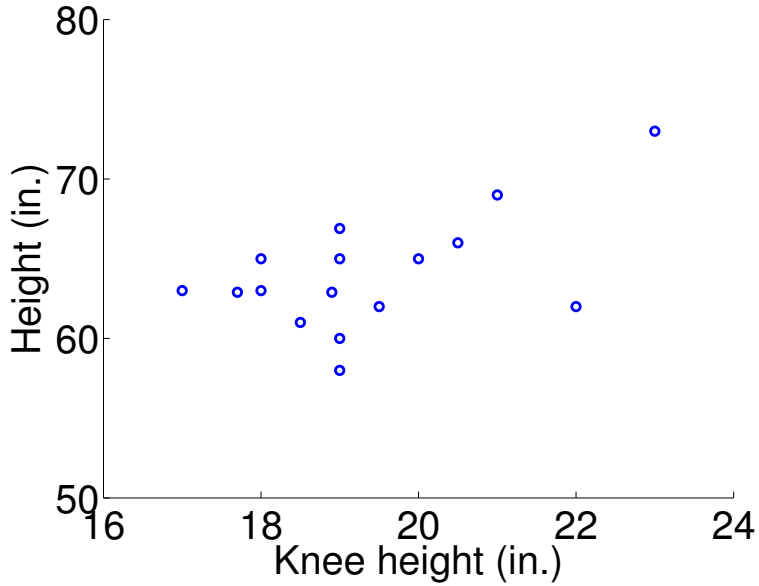
Supervised Learning

- Revisit Obama example from Lecture 1

Supervised Learning

- Your data:

Knee Height	Height
17	63
19	65
20.5	66
...	...



How can we predict height of a bedridden patient from knee height?

Supervised Learning

- Observe m "training examples" of form $(x^{(i)}, y^{(i)})$
  - $x^{(i)}$: **features** / input / what we observe / knee height
  - $y^{(i)}$: **target** / output / what we want to predict / height

- **Training set** $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Supervised Learning

- **Goal**: find function h such that $h(x)$ is a good predictor of output value
  - $y^{(i)} \approx h(x^{(i)})$ for training data
  - **Generalize** well to new x values

- Illustration on board: supervised learning

- Variations: type of x, y, h

Linear Regression in One Variable

- Let's consider our first example of supervised learning by assuming the hypothesis is a linear function:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

  - θ_0, θ_1: "parameters" or "weights"
  - θ_0: intercept, θ_1: slope

- How to find "best" θ_0, θ_1?

- Illustration: hypotheses

A Puzzle For You

- Let's assume $\theta_0 = 0$, so we only need to find θ_1
- **Tool**: you can minimize a function $J(\theta)$ of one variable by the following **optimization algorithm** from calculus:
  - Set $\frac{d}{dx} J(\theta) = 0$ and solve for θ
  - Illustration
- Can you devise an algorithm to find the "best" θ_1 using this tool?

Puzzle Answer: cost function

Use a **cost function** to measure the quality of a hypothesis (line) with slope θ_1. The "squared error" cost function is:

$$J(\theta_1) = \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Puzzle Answer: cost function

Use a **cost function** to measure the quality of a hypothesis (line) with slope θ_1. The "squared error" cost function is:

$$J(\theta_1) = \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

- E.g., $\theta_1 = 3$:

x	y	$(3x - y)^2$
17	63	$(51 - 63)^2 = 144$
19	65	$(57 - 65)^2 = 64$
20.5	66	$(61.5 - 65)^2 = 12.25$

$$J(3) = 144 + 64 + 12.25 = 220.25$$

Our First Algorithm

Find the hypothesis of minimum cost by setting the cost function derivative to zero and solving for θ_1. For this example:

$$\begin{aligned} J(\theta_1) &= (17 \cdot \theta_1 - 63)^2 + (19 \cdot \theta_1 - 65)^2 + (20.5 \cdot \theta_1 - 66)^2 \\ &= 1070.25 \cdot \theta_1^2 - 7318 \cdot \theta_1 + 12550 \end{aligned}$$

Our First Algorithm

Find the hypothesis of minimum cost by setting the cost function derivative to zero and solving for θ_1. For this example:

$$\begin{aligned} J(\theta_1) &= (17 \cdot \theta_1 - 63)^2 + (19 \cdot \theta_1 - 65)^2 + (20.5 \cdot \theta_1 - 66)^2 \\ &= 1070.25 \cdot \theta_1^2 - 7318 \cdot \theta_1 + 12550 \end{aligned}$$

$$0 = \frac{d}{d\theta_1} J(\theta_1) = 2140.5 \cdot \theta_1 - 7318$$

Our First Algorithm

Find the hypothesis of minimum cost by setting the cost function derivative to zero and solving for θ_1. For this example:

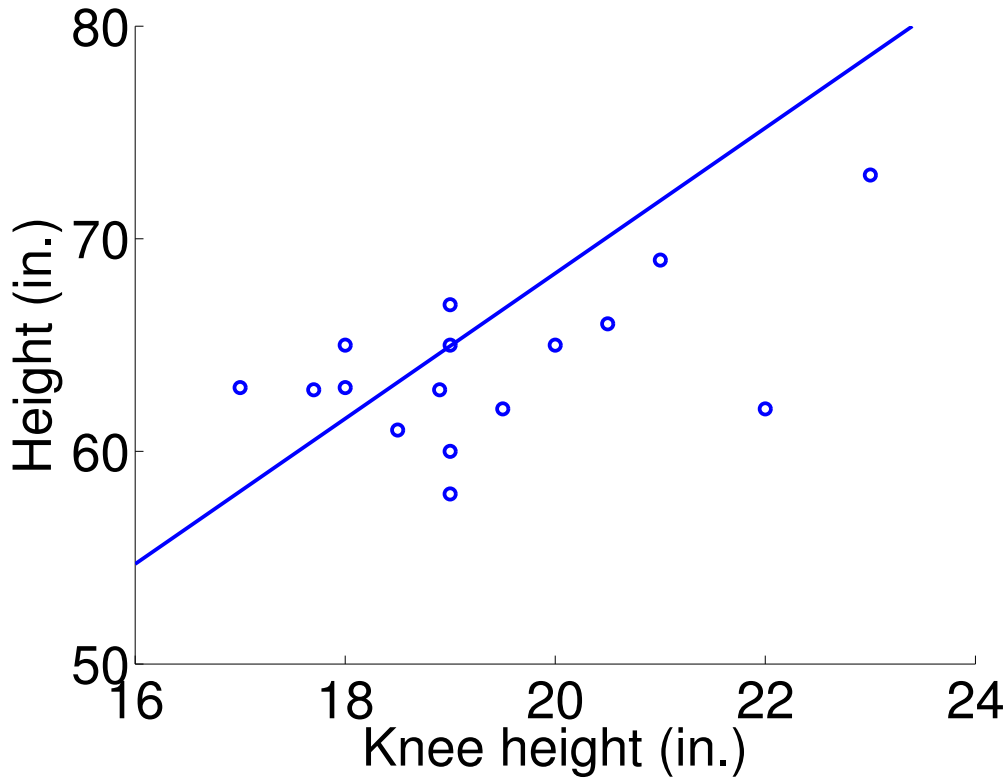
$$\begin{aligned} J(\theta_1) &= (17 \cdot \theta_1 - 63)^2 + (19 \cdot \theta_1 - 65)^2 + (20.5 \cdot \theta_1 - 66)^2 \\ &= 1070.25 \cdot \theta_1^2 - 7318 \cdot \theta_1 + 12550 \end{aligned}$$

$$0 = \frac{d}{d\theta_1} J(\theta_1) = 2140.5 \cdot \theta_1 - 7318$$

$$\theta_1 = \frac{7318}{2140.5} = 3.4188$$

(See `http://www.wolframalpha.com`)

Our First Algorithm In Action



The General Algorithm

Instead of plugging numbers into $J(\theta_1)$ and then finding the minimum, we can find the minimum in terms of $x^{(i)}$ and $y^{(i)}$

The general problem: find θ_1 to minimize

$$J(\theta_1) = \sum_{i=1}^{m} (\theta_1 x^{(i)} - y^{(i)})^2$$

You will solve this in HW1.

Two Problems Remain

- Problem one: we only fit the slope. What if $\theta_0 \neq 0$?
- Problem two: we will need a better optimization algorithm than "Set $\frac{d}{d\theta} J(\theta) = 0$ and solve for θ."
  - Wiggly functions
  - Equation(s) may be non-linear, hard to solve

Exercise: ideas for problem one?

Solution to Problem One

Design a cost function that takes two parameters:

$$\begin{aligned} J(\theta_0, \theta_1) &= \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2 \end{aligned}$$

Find θ_0, θ_1 to minimize $J(\theta_0, \theta_1)$

Functions of multiple variables

Here is an example cost function:

$$J(\theta_0, \theta_1) = (\theta_0 + 17 \cdot \theta_1 - 63)^2 + (\theta_0 + 19 \cdot \theta_1 - 65)^2 \\ + (\theta_0 + 20.5 \cdot \theta_1 - 66)^2 + (\theta_0 + 18.9 \cdot \theta_1 - 62.9)^2 + \dots$$

Gain intuition on `http://www.wolframalpha.com`

- Surface plot
- Contour plot

Solution to Problem Two: Gradient Descent

- **Gradient descent** is a general purpose optimization algorithm
- Idea: repeatedly take steps in steepest downhill direction, with step length proportional to "slope"
- Illustration: contour plot and pictorial definition of gradient descent

Gradient Descent

- Intialize θ_0, θ_1 arbitrarily
- Repeat until convergence

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

  - $\alpha =$ step-size (not too big)

Partial derivatives!

- The **partial derivative with respect to** θ_j is denoted $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
- Treat all other variables as constants, then take derivative
- Example

$$\begin{aligned} \frac{\partial}{\partial u} 5u^2 v^3 &= (5v^3)u^2 \\ &= 5v^3 \cdot 2u \\ &= 10v^3 u \end{aligned}$$

Partial derivatives!

- The **partial derivative with respect to** θ_j is denoted $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
- Treat all other variables as constants, then take derivative
- Example

$$\begin{aligned} \frac{\partial}{\partial u} 5u^2v^3 &= (5v^3)u^2 \\ &= 5v^3 \cdot 2u \\ &= 10v^3u \end{aligned}$$

$$\frac{\partial}{\partial v} 5u^2v^3 = ??$$

Partial derivative intuition

- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the rate of change along the θ_j axis
- Example: elliptical contours
  - Sign of $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$?
  - Sign of $\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$?
  - Which has larger absolute value?

Gradient Descent

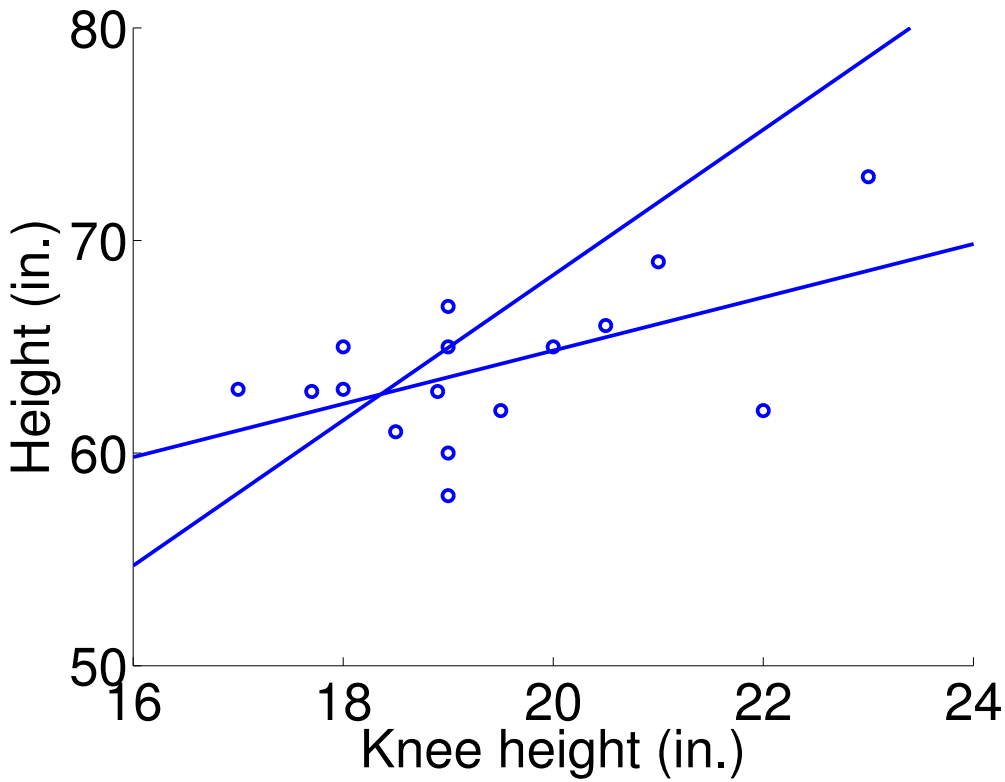
- Repeat until convergence

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

- Issues (explore in HW1)
  - How to set step-size α?
  - How to diagnose convergence?

The Result in Our Problem



$$h_\theta(x) = 39.75 + 1.25x$$

General gradient descent (for future topics)

- To minimize a function $J(\theta) := J(\theta_0, \theta_1, \ldots, \theta_n)$ of many variables:
  - Intialize θ_j arbitrarily for all j
  - Repeat until convergence

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_0} J(\theta) \quad \text{for all } j$$

(simultaneuous updates)

Summary

- What to know
  - Supervised learning setup
  - Cost function
    - Convert a learning problem to an optimization problem
    - Squared error
  - Gradient descent
- Next time
  - More on gradient descent
  - Linear algebra review