Linear Regression

Dan Sheldon

A First Supervised Learning Problem

How do you measure the biomass of a forest?

Hard to measure:

- Mass of tree
- Height of tree (but can be done)

Easy to measure:

- Diameter at breast height (DBH)

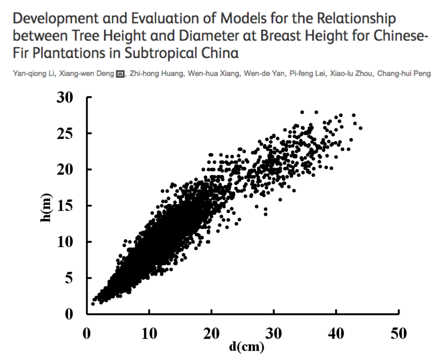
Let's simplify the problem: devise method to easily estimate the height of a tree

A First Supervised Learning Problem

Idea?

- Collect data on DBH and height for some trees
- Determine relationship between DBH and height
- Use DBH to predict height for a new tree

Some data



What do you predict for the height of a tree with DBH 15cm? 35cm? Why?

A First Supervised Learning Problem

Idea:

- Collect data on DBH and height for some trees
- Determine relationship between DBH and height
- Use DBH to predict height for a new tree

This is **supervised learning**:

- Collect **training data**
- Use a **learning algorithm** to fit a **model**
- Use model to make a **prediction**

What model? What algorithm? **Largely what this class is about.**

Supervised Learning

DBH (x)	Height (y)
17	63
19	65
20.5	66
...	...

Find h such that $h(x) \approx y$

Illustration on board: supervised learning

Supervised Learning: Notation and Terminology

- Observe m "training examples" of form $(x^{(i)}, y^{(i)})$
  - $x^{(i)}$: **features** / input / what we observe / DBH
  - $y^{(i)}$: **target** / output / what we want to predict / height
  - **Training set** $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$
- Find function ("hypothesis") h such that $h(x) \approx y$
  - $h(x^{(i)}) \approx y^{(i)}$ – good fit on training data
  - **Generalize** well to new x values

Variations: type of x, y, h

Linear Regression in One Variable

First example of supervised learning. Assume hypothesis is a linear function:

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

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

How to find "best" θ_0, θ_1? Illustration: hypotheses.

Finding the best hypothesis

Simplification: "slope-only" model $h_\theta(x) = \theta_1 x$

- We only need to find θ_1

Idea: design **cost function** $J(\theta_1)$ to numerically measure the quality of hypothesis $h_\theta(x)$

Exercise: which cost functions below make sense?

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

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

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

1. A only
2. B only
3. C only
4. B and C
5. A, B, and C

Answer. 4

Squared Error Cost Function

The "squared error" cost function is:

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

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

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

$$J(3) = (144 + 64 + 12.25)/2 = 220.25/2$$

Our First Algorithm

We can use calculus to find the hypothesis of minimum cost. Set the derivative of J to zero and solve for θ_1. For this example:

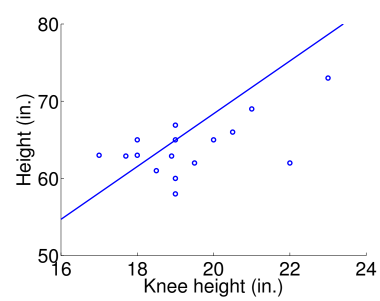
$$J(\theta_1) = \frac{1}{2}\Big((17\cdot\theta_1 - 63)^2 + (19\cdot\theta_1 - 65)^2 + (20.5\cdot\theta_1 - 66)^2\Big)$$
$$= 535.125\cdot\theta_1^2 - 3659\cdot\theta_1 + 6275$$

$$0 = \frac{d}{d\theta_1}J(\theta_1) = 1070.25\cdot\theta_1 - 3659$$

$$\theta_1 = \frac{3659}{1070.25} = 3.4188$$

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

Our First Algorithm In Action



The General Algorithm

In general, we don't want to plug numbers into $J(\theta_1)$ and solve a calculus problem *every time*.

Instead, we can solve for θ_1 in terms of $x^{(i)}$ and $y^{(i)}$.

The general problem: find θ_1 to minimize

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

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:

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

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

Functions of multiple variables!

Here is an example cost function:

$$\begin{aligned} J(\theta_0, \theta_1) = &\tfrac{1}{2}(\theta_0 + 17 \cdot \theta_1 - 63)^2 + \tfrac{1}{2}(\theta_0 + 19 \cdot \theta_1 - 65)^2 \\ &+ \tfrac{1}{2}(\theta_0 + 20.5 \cdot \theta_1 - 66)^2 + \tfrac{1}{2}(\theta_0 + 18.9 \cdot \theta_1 - 62.9)^2 + \ldots \end{aligned}$$

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. A "workhorse" of ML.
- 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

To minimize a function $J(\theta_0, \theta_1)$ of two variables

- Intialize θ_0, θ_1 arbitrarily
- Repeat until convergence

$$\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \theta_1 &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \end{aligned}$$

  - $\alpha =$ step-size or **learning rate** (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

$$\frac{\partial}{\partial u} 5u^2v^3 = 5v^3 \frac{\partial}{\partial u} u^2 = 5v^3 \cdot 2u = 10v^3 u$$

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

Partial derivative intuition

Interpretation of partial derivative: $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the rate of change along the θ_j axis

Example: illustrate funciton with 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?

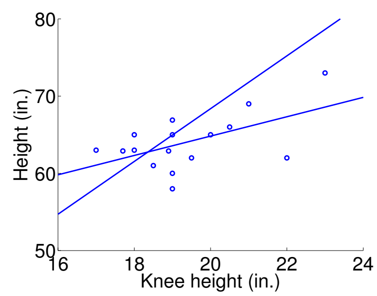
Gradient Descent

- 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)
  - Pitfalls
  - How to set step-size α?
  - How to diagnose convergence?

The Result in Our Problem



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

Gradient descent intuition

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

- Why does this move in the direction of steepest descent?
- What would we do if we wanted to maximize $J(\theta_0, \theta_1)$ instead?

Gradient descent for linear regression

Algorithm

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{for } j = 0, 1$$

Cost function

$$J(\theta_0, \theta_1) = \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$$

We need to calculate partial derivatives.

Linear regression partial derivatives

Let's first do this with a single training example (x, y):

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \big(h_\theta(x) - y\big)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \Big(\theta_0 + \theta_1 x - y\Big)\end{aligned}$$

So we get

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = (h_\theta(x) - y)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = (h_\theta(x) - y)x$$

Linear regression partial derivatives

More generally, with many training examples (work this out):

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

So the algorithm is:

$$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

Demo: parameter space vs. hypotheses

Show gradient descent demo

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