

L10 Linear Regression on Real Datasets


Prof. Xun Jiao

WINNERS



ARTIFICIAL INTELLIGENCE

Amazon's New A.I. Tool Will Launch for the Super Bowl, But It Could Change Much More Than Football Artificial

intelligence technology that can measure quarterback play is expected to have applications across a variety of workplaces. 

<https://www.inc.com/kevin-j-ryan/super-bowl-amazon-machine-learning-passer-score-artificial-intelligence.html>

The NFL is unveiling a new measure of quarterbacks' performances within a given game.

Called the Passing Score, the metric uses Amazon Web Services' A.I. to assess a quarterback's execution on each play. At the end of the game, the player is graded on a scale of 50 to 99. Priya Ponnappalli, senior manager and principal scientist at the Amazon Machine Learning Solutions Lab, says the tool can have implications beyond the sports world.

Computer chips embedded in players' equipment, on-field pylons, and footballs during games track the players' location, movement, and speed throughout the game, Ponnappalli says. That data can be fed into a series of machine-learning models to come up with new metrics like the Passing Score.

Engineers, analysts, and data scientists at Next Gen Stats, the NFL's advanced statistics arm, spent nearly a year applying A.I. and other data analysis tools from AWS to build the metric. "There was no existing playbook to build this kind of tool," says Ponnappalli.

The team combined seven different machine-learning models to create the Passing Score. Among them is a new tool that evaluates the positioning and movement of the players to predict the likelihood a pass will be completed before it is thrown. This allows the NFL to measure, among other things, the most impressive plays that a quarterback made, which could be useful for coaches when reviewing game footage.

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \quad \begin{array}{l} \text{simultaneous} \\ \text{update} \\ \text{for } j = 0 \dots d \end{array}$$

- To achieve simultaneous update
 - At the start of each GD iteration, compute $h_{\theta} \left(\mathbf{x}^{(i)} \right)$
 - Use this stored value in the update step loop
- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

$$\text{L}_2 \text{ norm: } \quad \|\mathbf{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \dots + v_{|\mathbf{v}|}^2}$$

Solve real problems

- How to train a predictor for beer rating?

Beeradvocate

Beers:



BA SCORE
100
world-class
9,587 Ratings

THE BROS
95
world-class
(view ratings)

Ratings: 9,587
Reviews: 2,537
rAvg: 4.59
pDev: 9.59%
Wants: 2,109
Gots: 4,563 | FT: 472

Brewed by:
Goose Island Beer Co. 
Illinois, United States


Style | ABV
American Double / Imperial Stout | 13.80% ABV

Availability: Winter

Notes/Commercial Description:
60 IBU
(Beer added by: drewbage on 06-26-2003)

Displayed for educational use only;
do not reuse.

Ratings/reviews:



4.35/5 rDev -5.2%
look: 4 | smell: 4.25 | taste: 4.5 | feel: 4.25 | overall: 4.25

Serving: 355 mL bottle poured into a 9 oz Libbey Embassy snifter ("bottled on: 08AUG14 1109").

Appearance: Deep, dark near-black brown. Hazy, light brown fringe of foam and limited lacing; no head.

Smell: Roasted malt, vanilla, and some warming alcohol.

Taste: Roasted malts, cocoa, burnt caramel, molasses, vanilla and dark fruit. Bourbon barrel is hinted at but never takes over.

Mouthfeel: Medium to full body and light carbonation with a very lush, silky smooth feel.

Overall: Not as complex or intense as some newer barrel-aged stouts, but so smooth and balanced with all the elements tightly integrated.

HipCzech, Yesterday at 05:38 AM

User profiles:



HipCzech
Aficionado
Male, from Texas

Profile Page

Member Since: **Jul 12, 2014** HipCzech was last seen:
Points: **175** Today at 12:19 AM
Beers: **108**
Places: **6**
Posts: **0**
Likes Received: **0**
Trading: **0% | 0**

50,000 beer rating reviews are available on

- https://www.ece.villanova.edu/~xjiao/course/ECE5400/dataset/beer_50000.json

Example 1

- Using ordinary linear regression, train a predictor that uses the age ('ageInSeconds') to predict the beer rating ('review/overall'), i.e.,
 - $\text{review/overall} = \theta_0 + \theta_1 * \text{ageInSeconds}$
 - You may use Python libraries to do so. What are the fitted values of θ_0 and θ_1 ?


```
import numpy
import urllib
import urllib.request
import scipy.optimize
import random

def parseData(fname):
    for l in urllib.request.urlopen(fname):
        yield eval(l)

print ("Reading data")
data = list(parseData("https://www.ece.villanova.edu/~xjiao/course/ECE5400/dataset/beer_50000.json"))
print ("done")

### Do older people rate beer more highly? ###

data2 = [d for d in data if 'user/ageInSeconds' in d]

def feature(datum):
    feat = [1]
    feat.append(datum['user/ageInSeconds'])
    return feat

X = [feature(d) for d in data2]
y = [d['review/overall'] for d in data2]
theta, residuals, rank, s = numpy.linalg.lstsq(X, y)
print(theta)

~
```

Check yourself

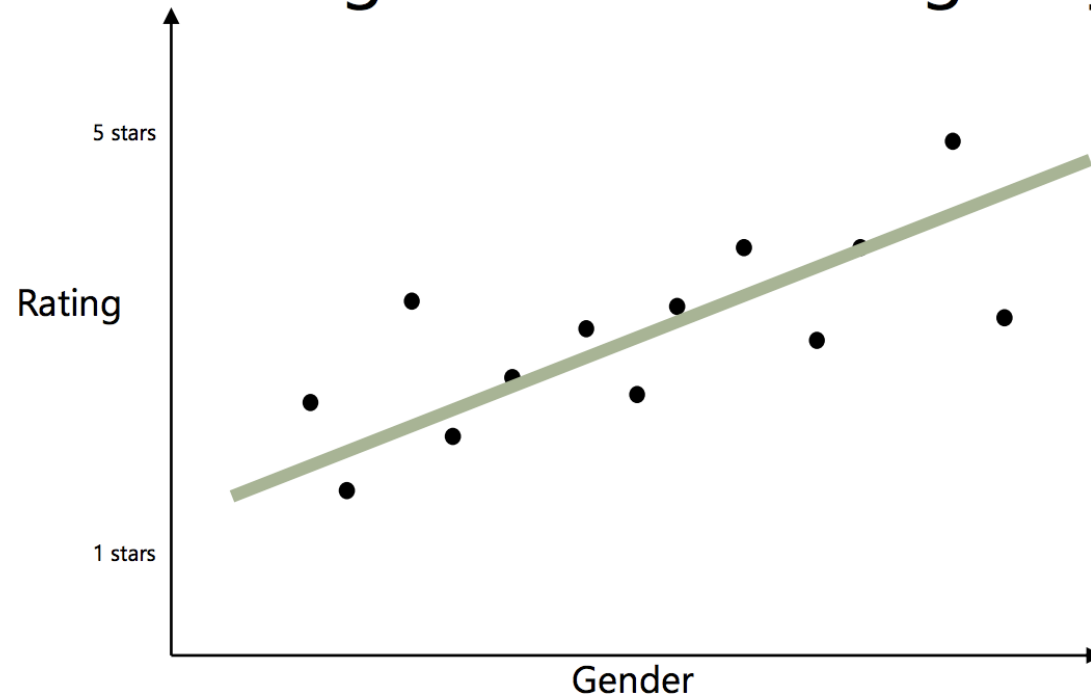
- What if I want to predict rating of beer using ABV?

Example 2

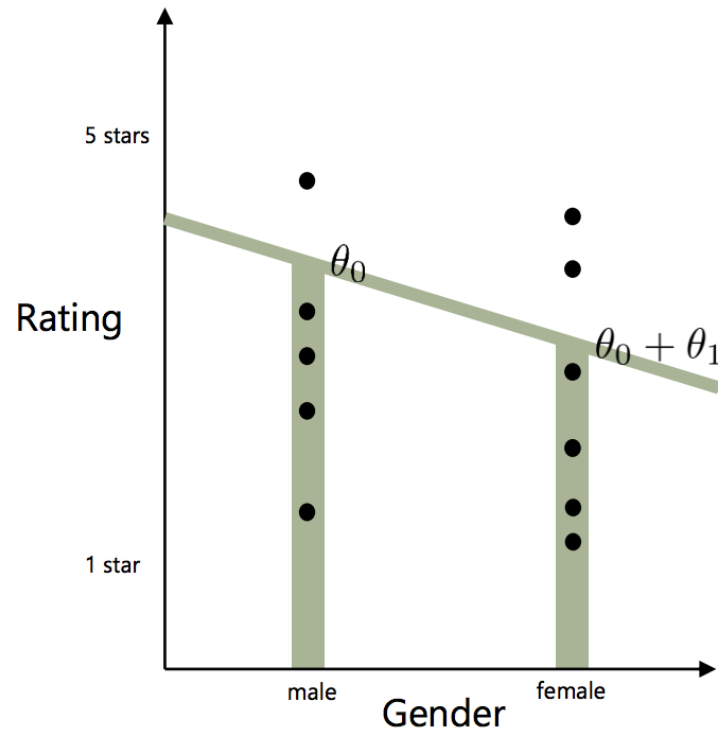
- We have used age, ABV as our feature.
 - These features are real-value numbers
- What if I want to use gender as my feature?
 - How to represent gender as feature?

Do we still have sth like this?

E.g. How does rating vary with **gender**?



It should be like this



θ_0 is the (predicted/average) rating for males

θ_1 is the **how much higher** females rate than males (in this case a negative number)

We're really still fitting a line though!

What if we had more than two values?
(e.g {"male", "female", "other", "not specified"})

Could we apply the same approach?

$$\text{Rating} = \theta_0 + \theta_1 \times \text{gender}$$

gender = **0 if "male", 1 if "female", 2 if "other", 3 if "not specified"**

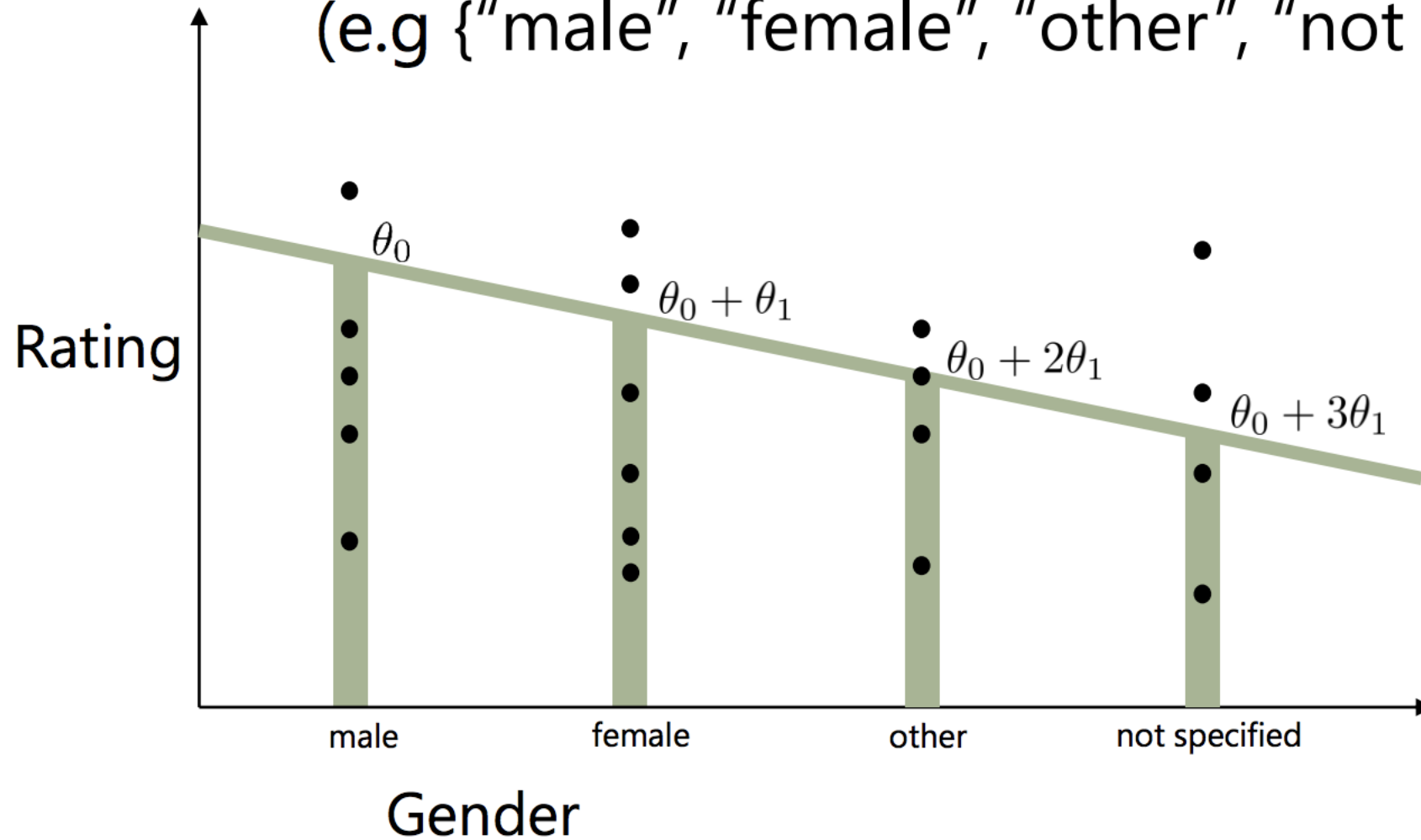
$$\text{Rating} = \theta_0 \quad \textbf{if male}$$

$$\text{Rating} = \theta_0 + \theta_1 \quad \textbf{if female}$$

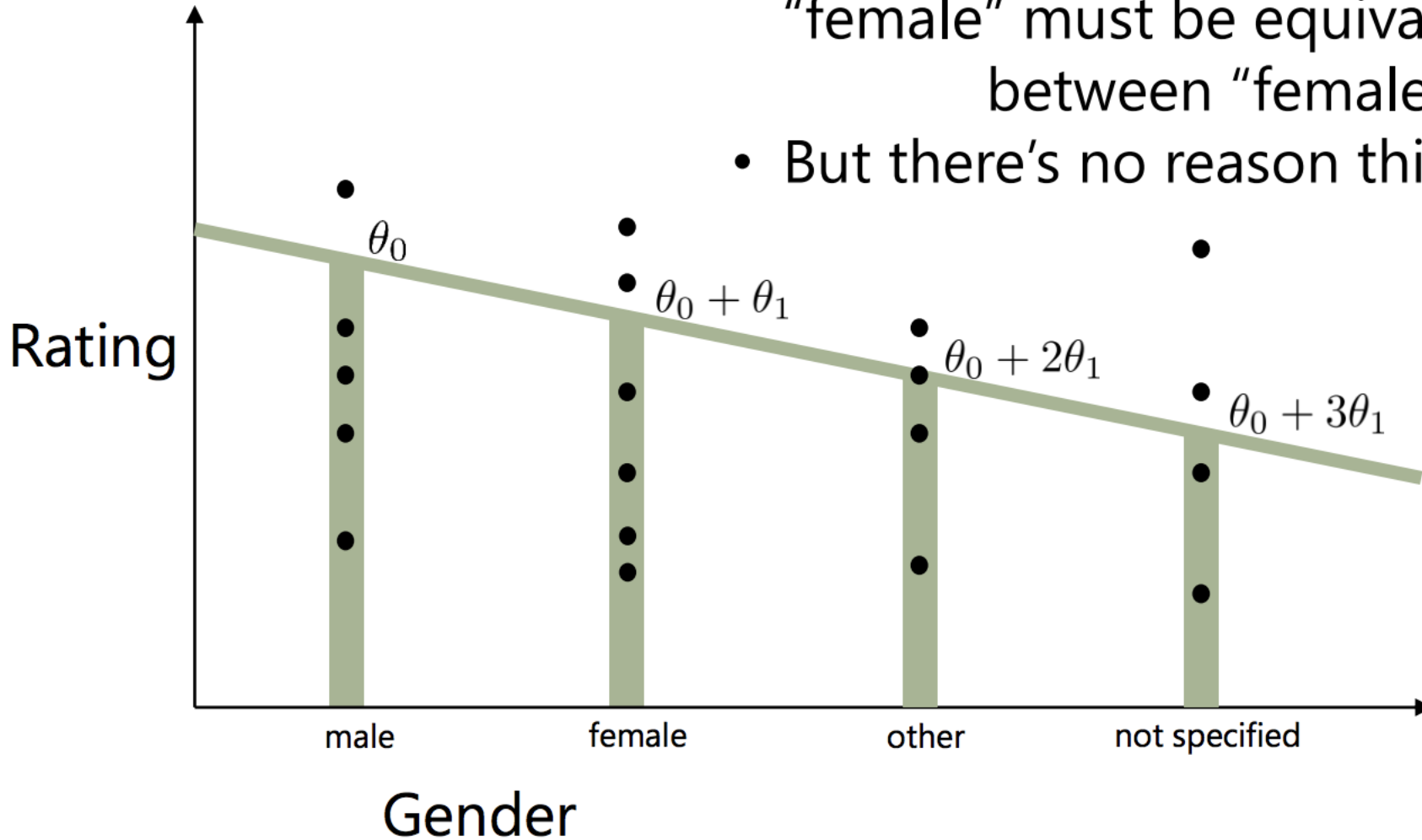
$$\text{Rating} = \theta_0 + 2\theta_1 \quad \textbf{if other}$$

$$\text{Rating} = \theta_0 + 3\theta_1 \quad \textbf{if not specified}$$

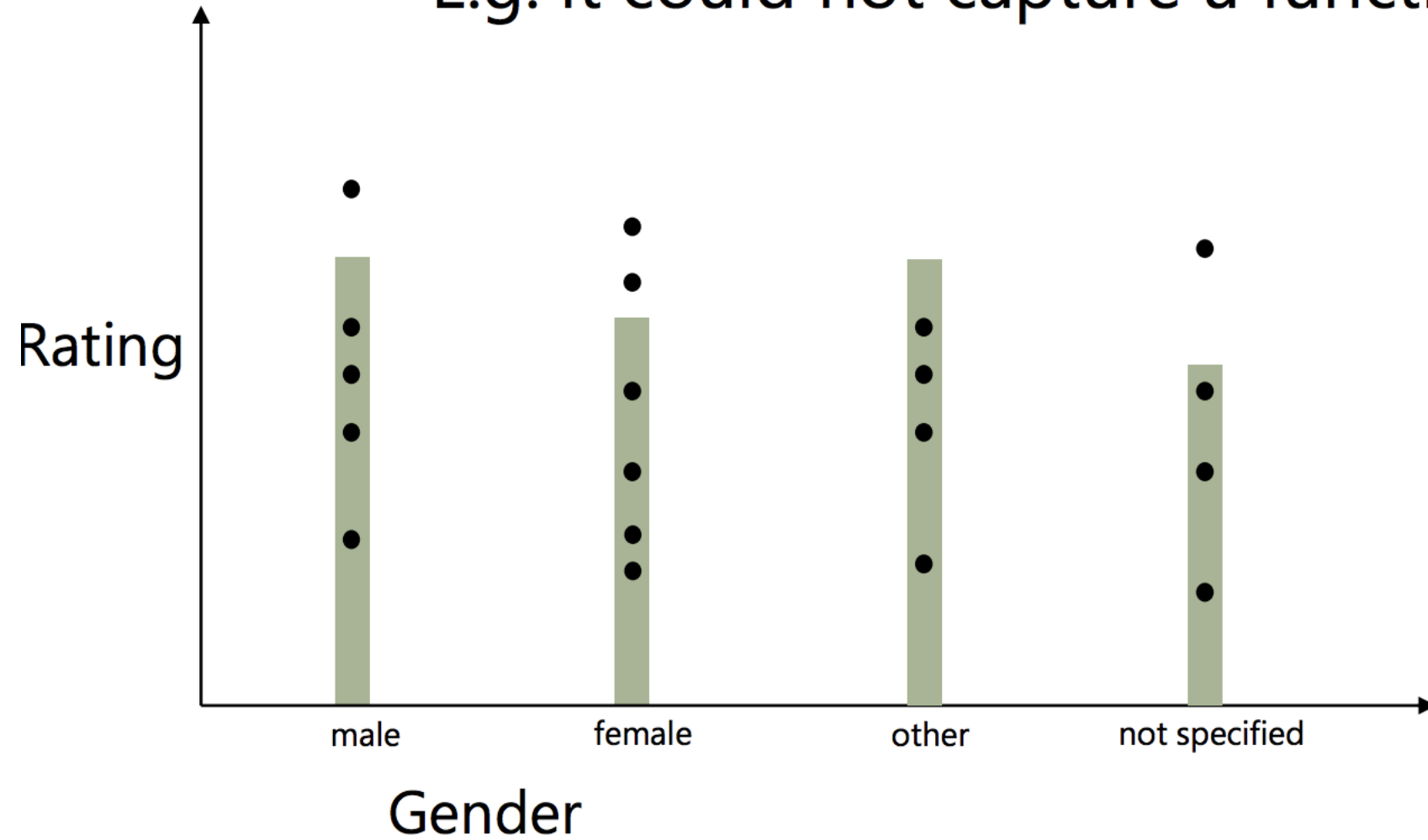
What if we had more than two values?
(e.g {"male", "female", "other", "not specified"})



- This model is **valid**, but won't be very **effective**
- It assumes that the difference between "male" and "female" must be equivalent to the difference between "female" and "other"
- But there's no reason this should be the case!



E.g. it could not capture a function like:



Instead we need something like:

$$\text{Rating} = \theta_0 \quad \textbf{if male}$$

$$\text{Rating} = \theta_0 + \theta_1 \quad \textbf{if female}$$

$$\text{Rating} = \theta_0 + \theta_2 \quad \textbf{if other}$$

$$\text{Rating} = \theta_0 + \theta_3 \quad \textbf{if not specified}$$

This is equivalent to:

$$(\theta_0, \theta_1, \theta_2, \theta_3) \cdot (1; \text{feature})$$

where feature = [1, 0, 0] for "female"
 feature = [0, 1, 0] for "other"
 feature = [0, 0, 1] for "not specified"

One-Hot Coding

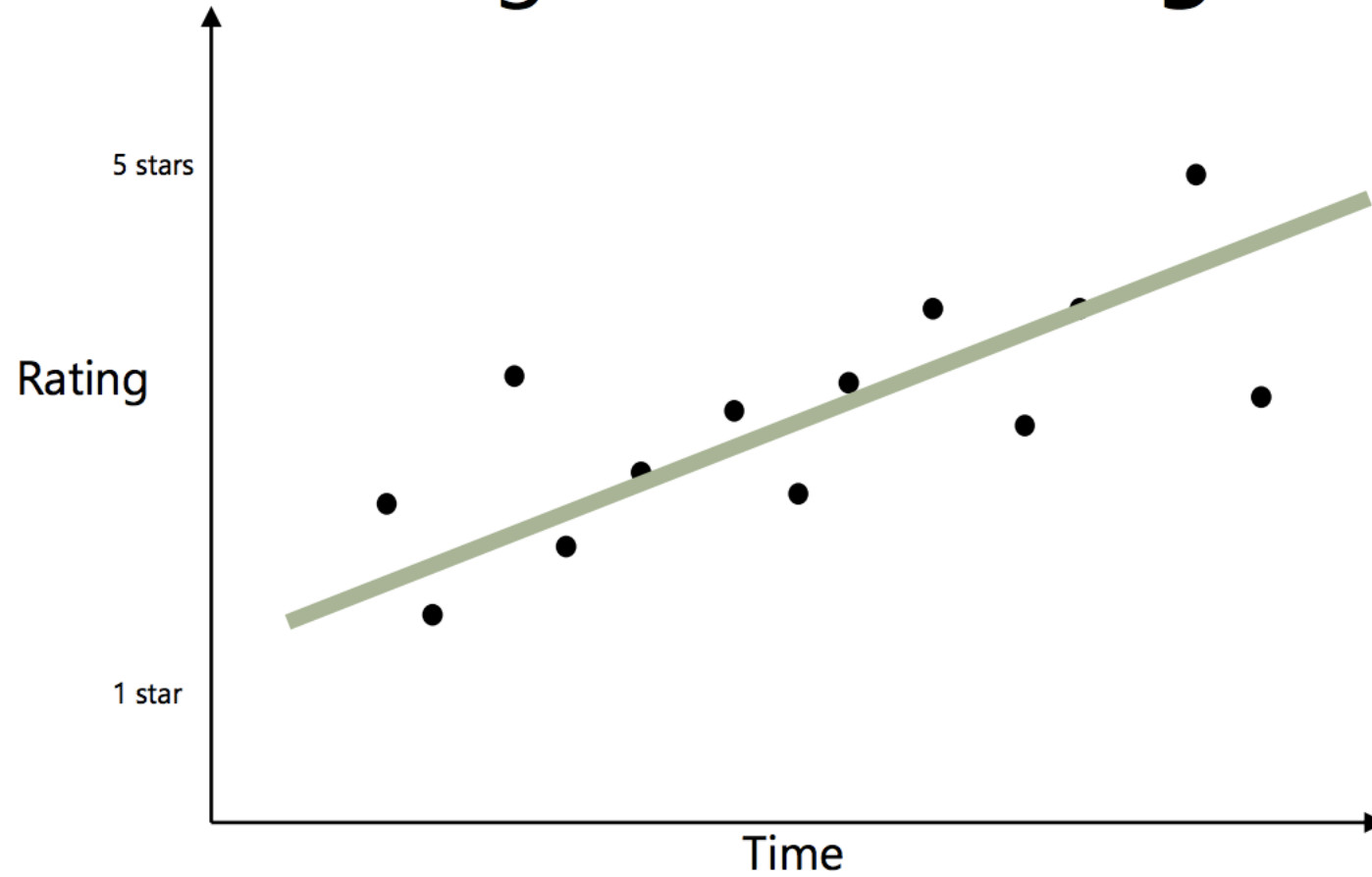
feature = [1, 0, 0] for "female"
feature = [0, 1, 0] for "other"
feature = [0, 0, 1] for "not specified"

- This type of encoding is called a **one-hot encoding** (because we have a feature vector with only a single "1" entry)
- Note that to capture 4 possible categories, we only need three dimensions (a dimension for "male" would be redundant)
- This approach can be used to capture a variety of categorical feature types, as well as objects that belong to multiple categories

Example 3

How would you build a feature to represent the **month**, and the impact it has on people's rating behavior?

E.g. How do **ratings** vary with **time**?



E.g. How do **ratings** vary with **time**?

- In principle this picture looks okay (compared our previous example on categorical features) – we're predicting a **real valued** quantity from **real valued** data (assuming we convert the date string to a number)
- So, what would happen if (e.g. we tried to train a predictor based on the month of the year)?

E.g. How do **ratings** vary with **time**?

- Let's start with a simple feature representation, e.g. map the month name to a month number:

$$\text{rating} = \theta_0 + \theta_1 \times \text{month} \quad \text{where}$$

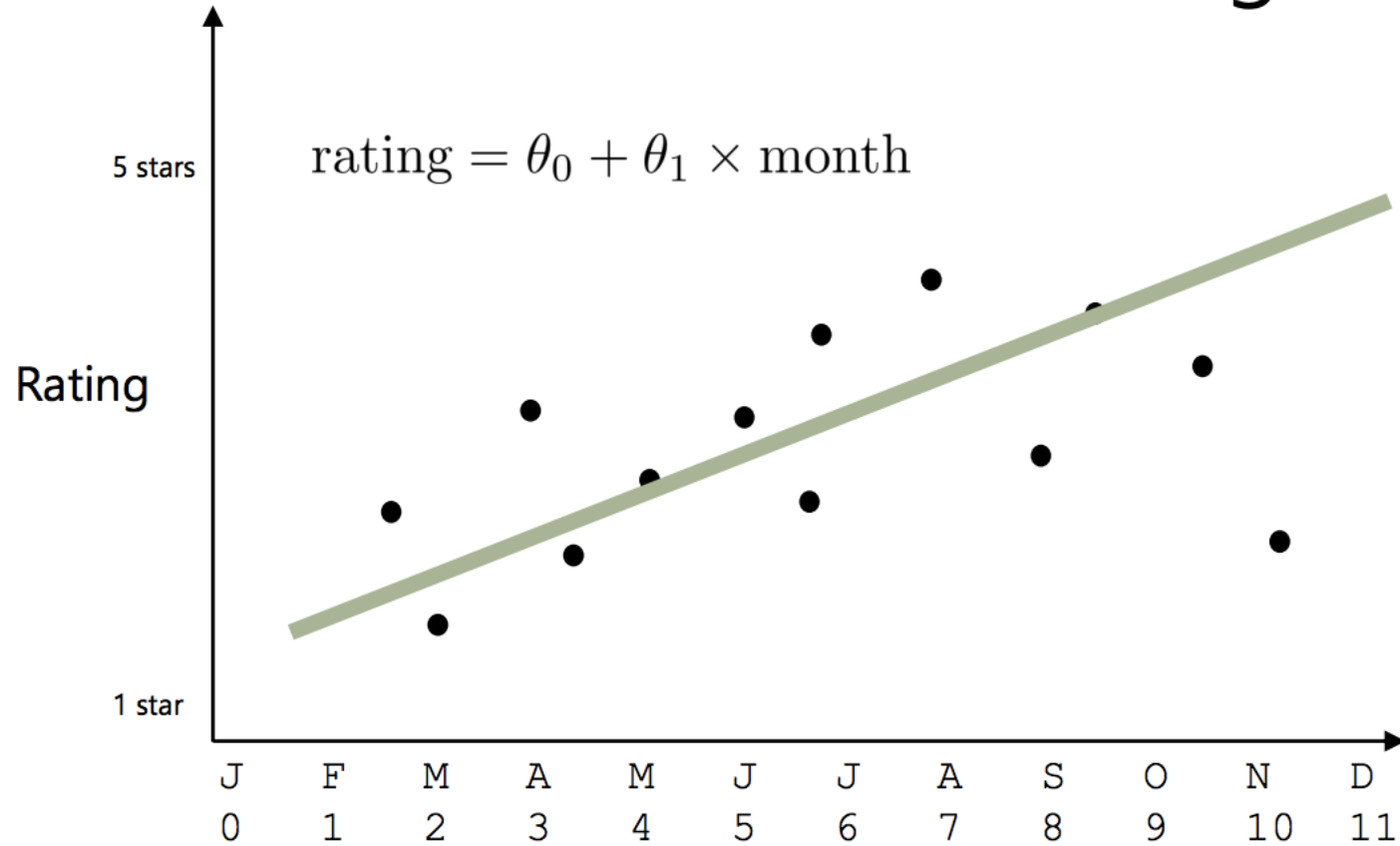
Jan = [0]

Feb = [1]

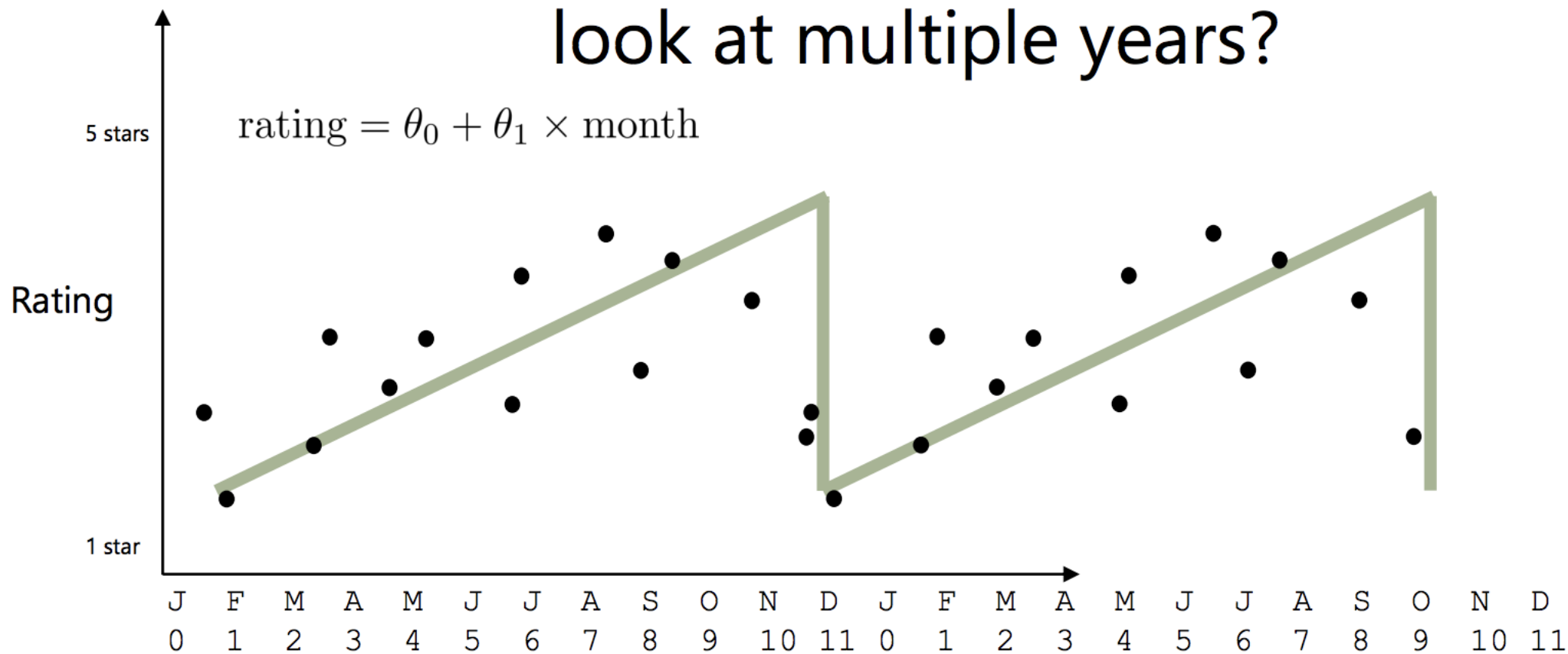
Mar = [2]

etc.

The model we'd learn might look something like:



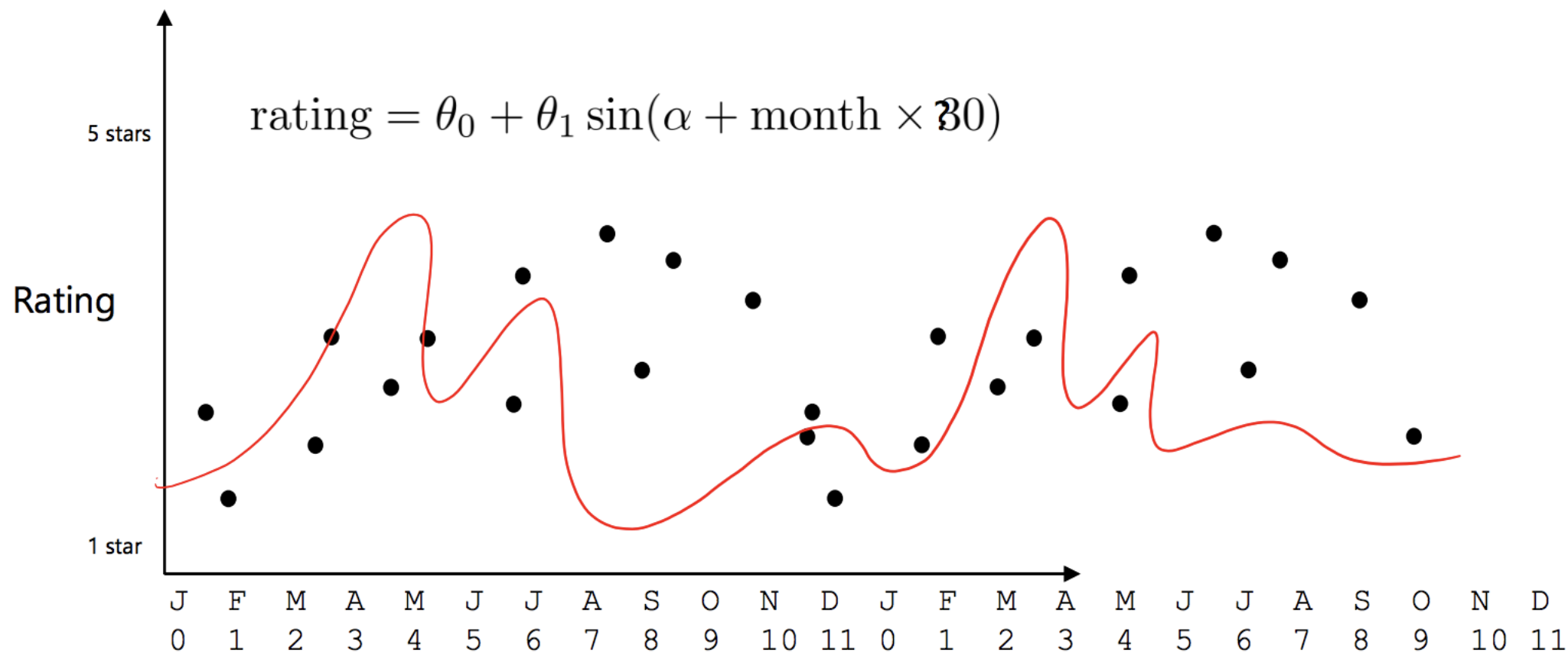
This seems fine, but what happens if we look at multiple years?



This seems fine, but what happens if we look at multiple years?

- This representation implies that the model would “wrap around” on December 31 to its January 1st value.
- This type of “sawtooth” pattern probably isn’t very realistic

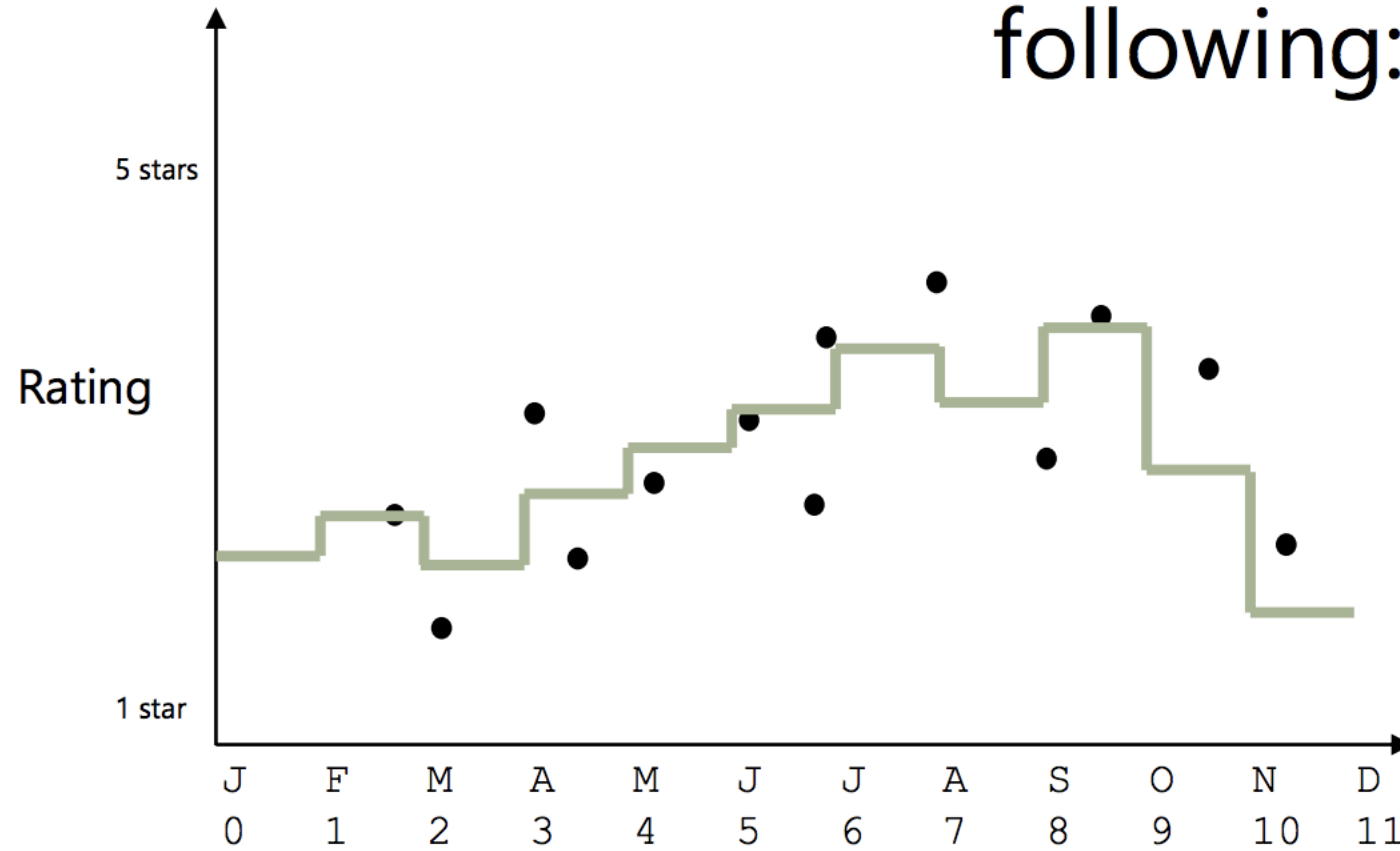
What might be a more realistic shape?



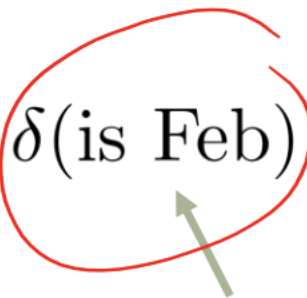
Fitting some periodic function like a sin wave would be a valid solution, but is difficult to get right, and fairly inflexible

- Also, it's not a **linear model**
- **Q:** What's a class of functions that we can use to capture a more flexible variety of shapes?
- **A:** Piecewise functions!

We'd like to fit a function like the following:



In fact this is very easy, even for a linear model! This function looks like:

$$\text{rating} = \theta_0 + \theta_1 \times \delta(\text{is Feb}) + \theta_2 \times \delta(\text{is Mar}) + \theta_3 \times \delta(\text{is Apr}) \dots$$


1 if it's Feb, 0
otherwise

- Note that we don't need a feature for January
- i.e., θ_0 captures the January value, θ_0 captures the *difference* between February and January, etc.

Or equivalently we'd have features as follows:

$$\text{rating} = \theta \cdot x \quad \text{where}$$

$x =$ $[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ if February
 $[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ if March
 $[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ if April
 \dots
 $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$ if December

Note that this is still a form of **one-hot** encoding, just like we saw in the “categorical features” example

- This type of feature is very flexible, as it can handle complex shapes, periodicity, etc.
- We could easily increase (or decrease) the resolution to a week, or an entire season, rather than a month, depending on how fine-grained our data was

We can also extend this by combining several one-hot encodings together:

$$\text{rating} = \theta \cdot x = \theta \cdot [x_1; x_2] \text{ where}$$

x_1 = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] if February
[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0] if March
[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0] if April
...
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1] if December

x_2 = [1, 0, 0, 0, 0, 0] if Tuesday
[0, 1, 0, 0, 0, 0] if Wednesday
[0, 0, 1, 0, 0, 0] if Thursday
...

Season vs.
rating (overall)

