

# CSE 158 – Lecture 2

Web Mining and Recommender Systems

Supervised learning – Regression

# Supervised versus unsupervised learning

**Learning** approaches attempt to **model data** in order to solve a problem

**Unsupervised learning** approaches find patterns/relationships/structure in data, but **are not** optimized to solve a particular predictive task

**Supervised learning** aims to directly model the relationship between input and output variables, so that the output variables can be predicted accurately given the input

# Regression

**Regression** is one of the simplest supervised learning approaches to learn relationships between input variables (features) and output variables (predictions)

# Linear regression

**Linear regression** assumes a predictor of the form

$$X\theta = y$$

matrix of features  
(data)

unknowns  
(which features are relevant)

vector of outputs  
(labels)

(or  $Ax = b$  if you prefer)

# Linear regression

**Linear regression** assumes a predictor of the form

$$X\theta = y$$

**Q:** Solve for theta

**A:**  $\theta = (X^T X)^{-1} X^T y$

# Example 1

## Beeradvocate

### Beers:



Displayed for educational use only; do not reuse.

BA SCORE <b>100</b> world-class 9,587 Ratings	THE BROS <b>95</b> 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)

### 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:	<b>Jul 12, 2014</b>	HipCzech was last seen:
Points:	<b>175</b>	Today at 12:19 AM
Beers:	<b>108</b>	
Places:	<b>6</b>	
Posts:	smoother than all of	<b>0</b>
Likes Received:	<b>0</b>	
Trading:	<b>0%   0</b>	

# Example 1

50,000 reviews are available on

[http://jmcauley.ucsd.edu/cse158/data/beer/beer\\_50000.json](http://jmcauley.ucsd.edu/cse158/data/beer/beer_50000.json)

(see course webpage)

See also – non-alcoholic beers:

<http://jmcauley.ucsd.edu/cse158/data/beer/non-alcoholic-beer.json>

# Example 1

## Real-valued features

How do preferences toward certain beers vary with age?

How about **ABV**?

(code for all examples is on <http://jmcauley.ucsd.edu/cse158/code/week1.py>)



# Example 1.5: Polynomial functions

What about something like  $ABV^2$ ?

$$\text{rating} = \theta_0 + \theta_1 \times ABV + \theta_2 \times ABV^2 + \theta_3 \times ABV^3$$

- Note that this is perfectly straightforward: the model still takes the form

$$\text{weight} = \theta \cdot x$$

- We just need to use the feature vector

$$x = [1, ABV, ABV^2, ABV^3]$$

# Fitting complex functions

Note that we can use the same approach to fit arbitrary functions of the features! E.g.:

$$\text{Rating} = \theta_0 + \theta_1 \times \text{ABV} + \theta_2 \times \text{ABV}^2 + \theta_3 \exp(\text{ABV}) + \theta_4 \sin(\text{ABV})$$

- We can perform arbitrary combinations of the **features** and the model will still be linear in the **parameters** (theta):

$$\text{Rating} = \theta \cdot x$$

# Fitting complex functions

The same approach would **not** work if we wanted to transform the parameters:

$$\text{Rating} = \theta_0 + \theta_1 \times \text{ABV} + \theta_2^2 \times \text{ABV} + \sigma(\theta_3) \times \text{ABV}$$

- The **linear** models we've seen so far do not support these types of transformations (i.e., they need to be linear in their parameters)
- There *are* alternative models that support non-linear transformations of parameters, e.g. neural networks

## Example 2

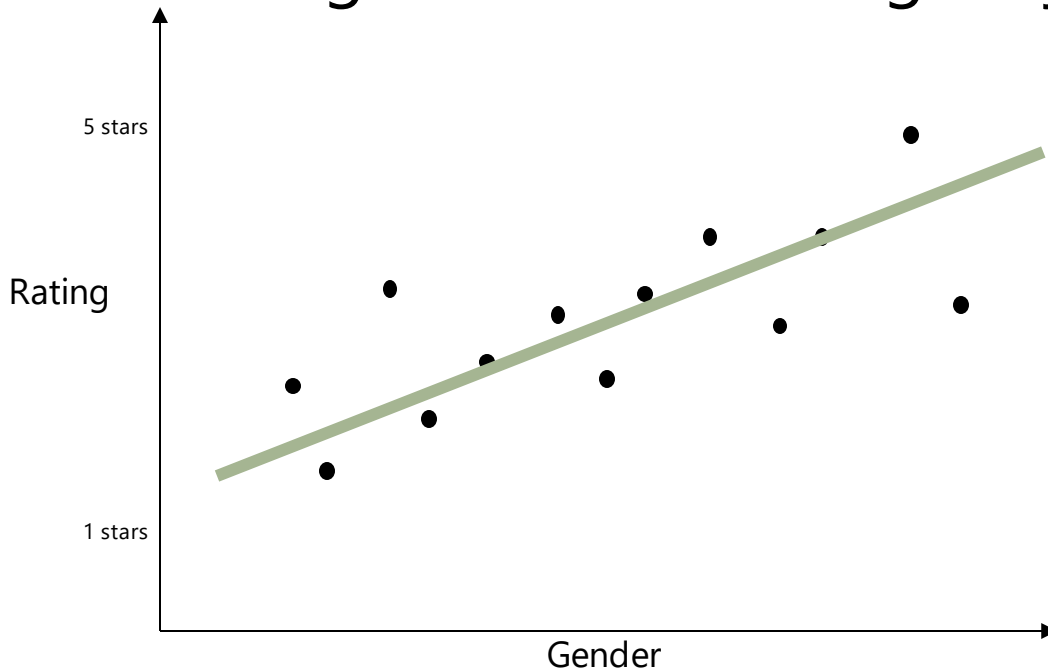
### Categorical features

How do beer preferences vary as a function of **gender**?

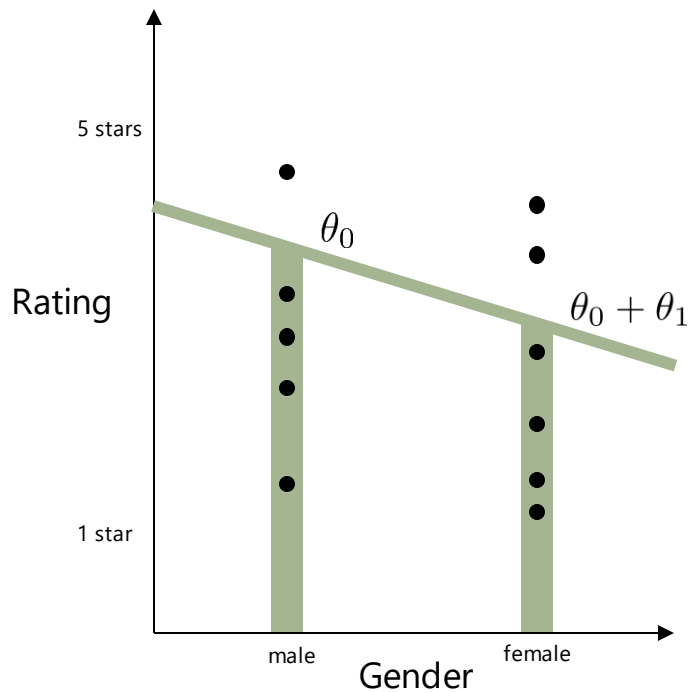
(code for all examples is on <http://jmcauley.ucsd.edu/cse158/code/week1.py>)

# Example 2

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



# Example 2



$\theta_0$  is the (predicted/average) rating for males

$\theta_1$  is the **how much higher** females rate than males (in this case a negative number)

We're really still fitting a line though!

# Motivating examples

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"**

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

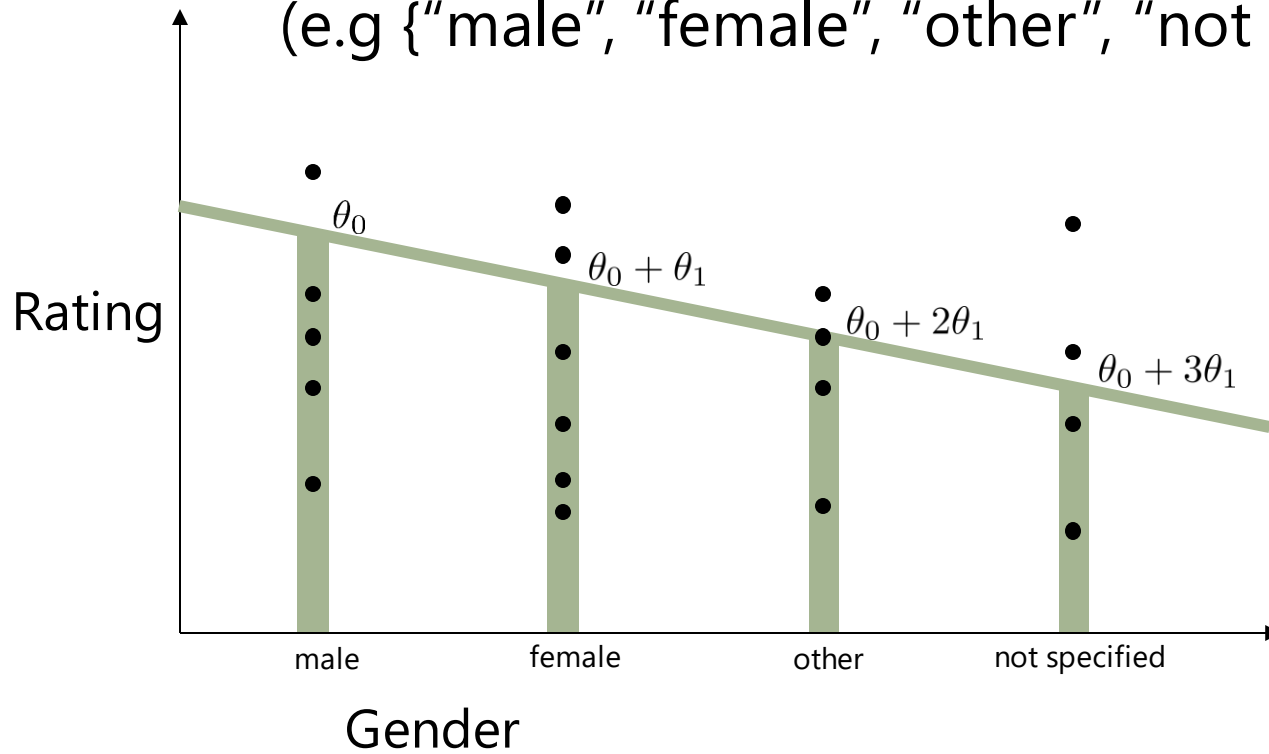
$$\text{Rating} = \theta_0 + \theta_1 \quad \mathbf{\text{if female}}$$

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

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

# Motivating examples

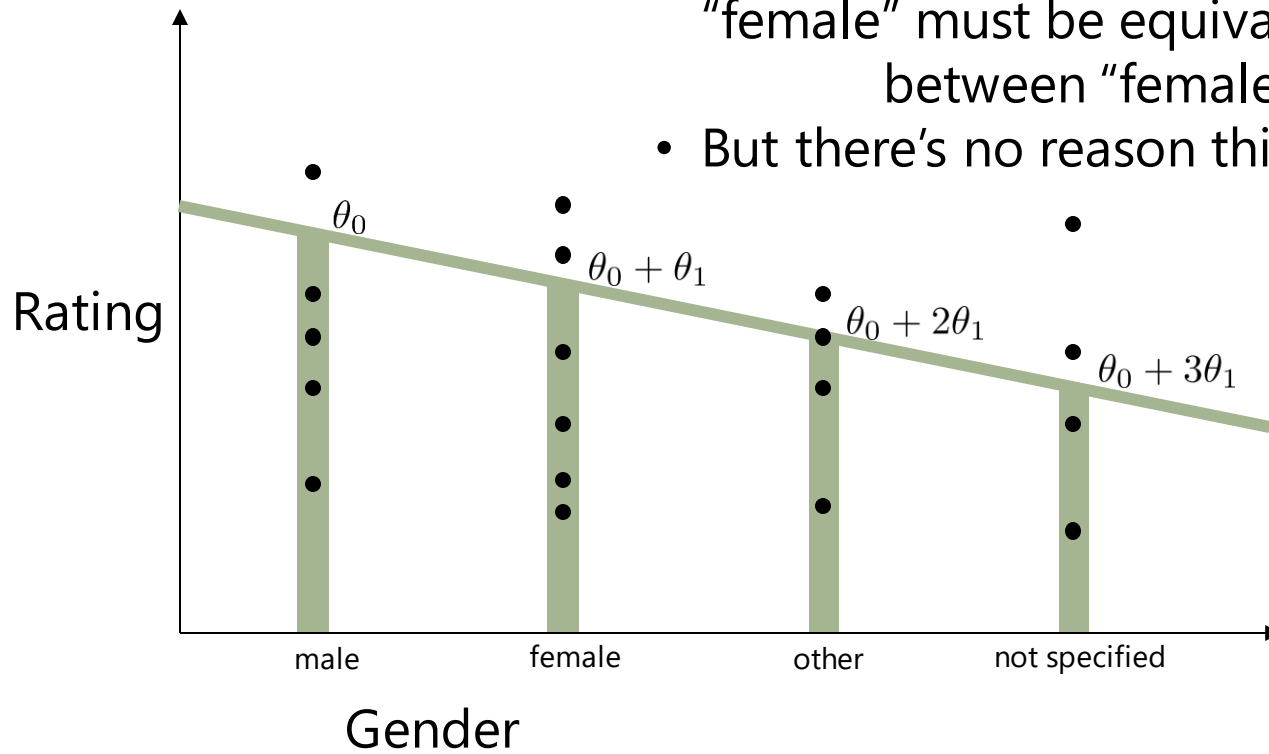
What if we had more than two values?  
(e.g {"male", "female", "other", "not specified"})





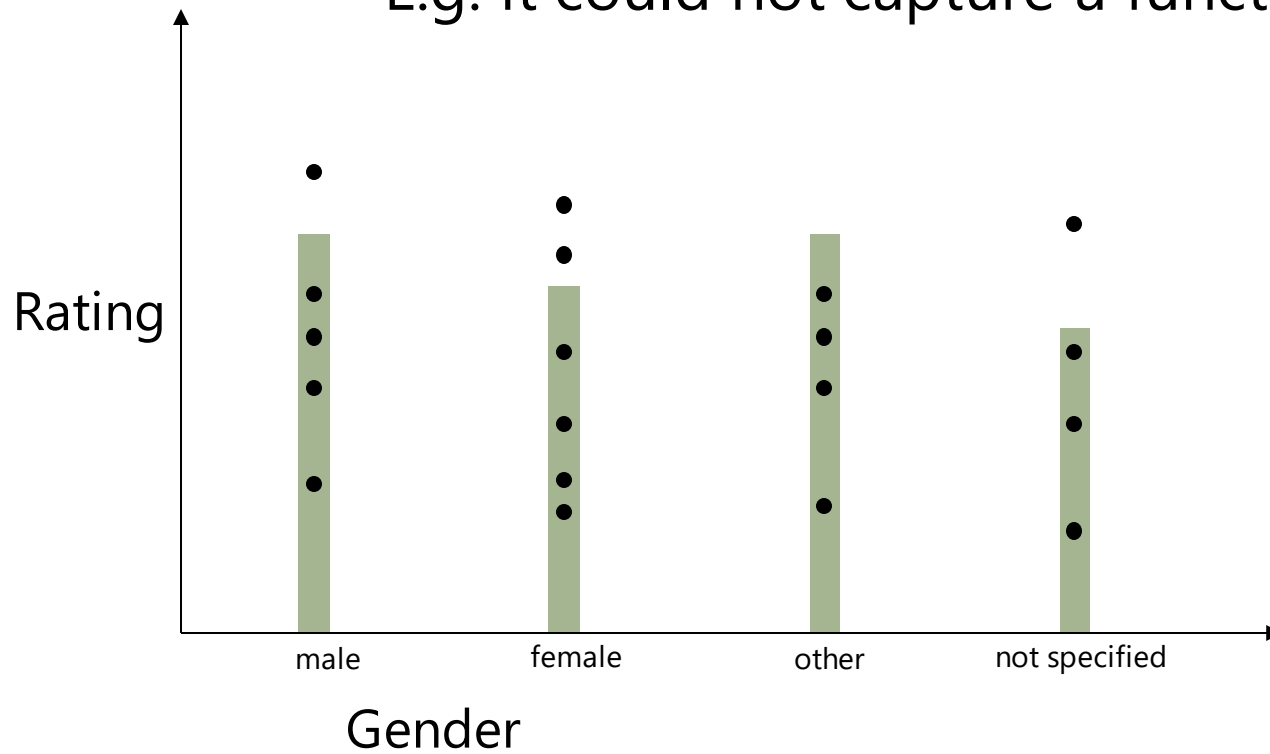
# Motivating examples

- 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!



# Motivating examples

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



# Motivating examples

Instead we need something like:

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

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

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

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

# Motivating examples

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"

# Concept: One-hot encodings

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

# Linearly dependent features

# Linearly dependent features

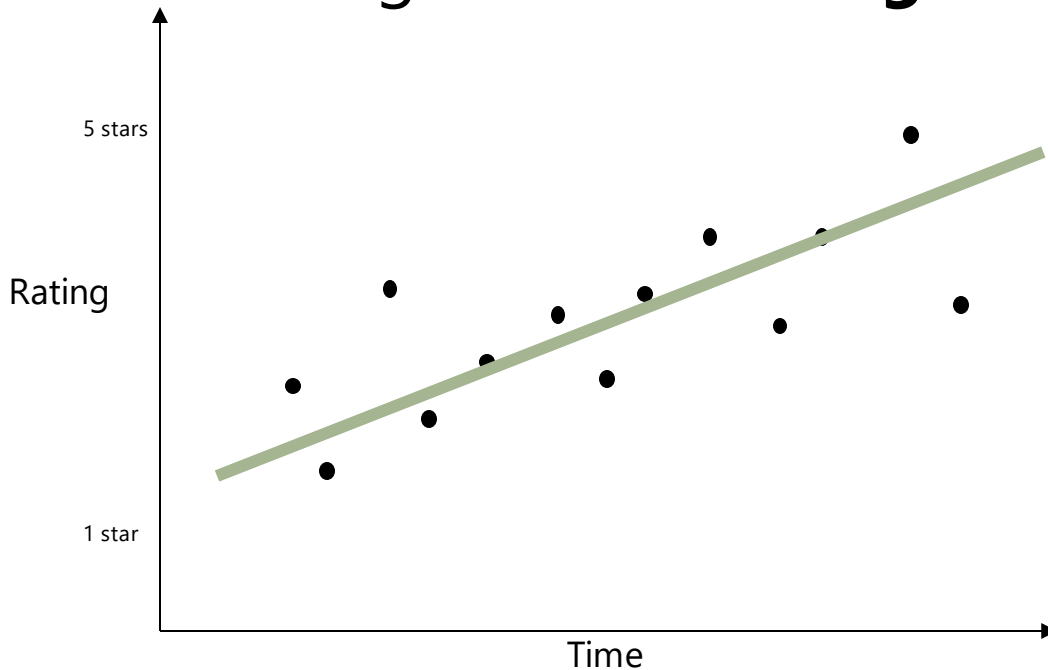
## Example 3

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



# Motivating examples

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



# Motivating examples

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)?

# Motivating examples

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]

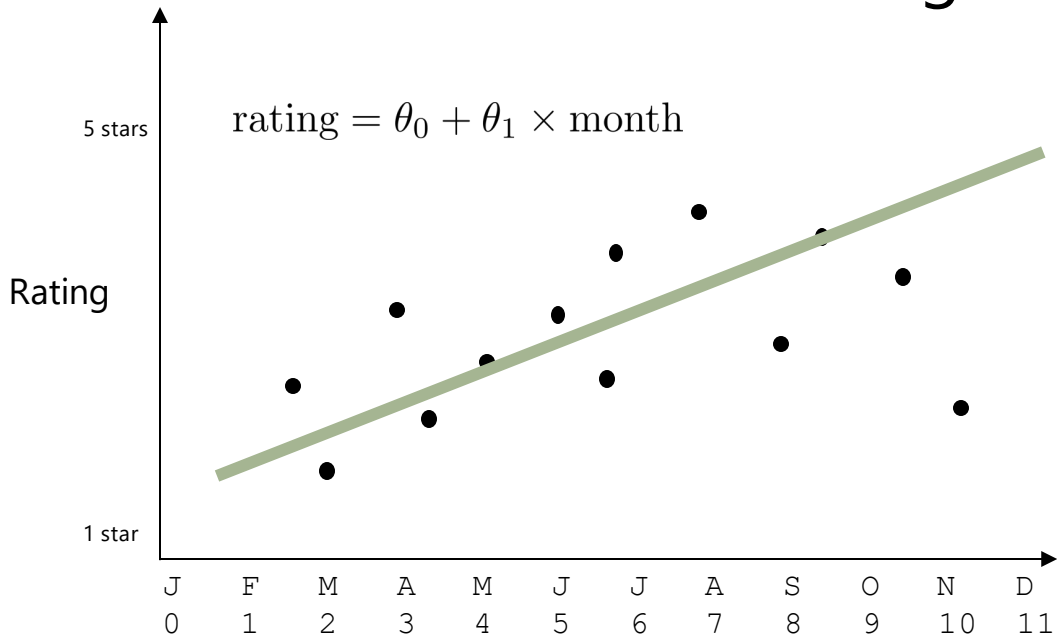
Feb = [1]

Mar = [2]

etc.

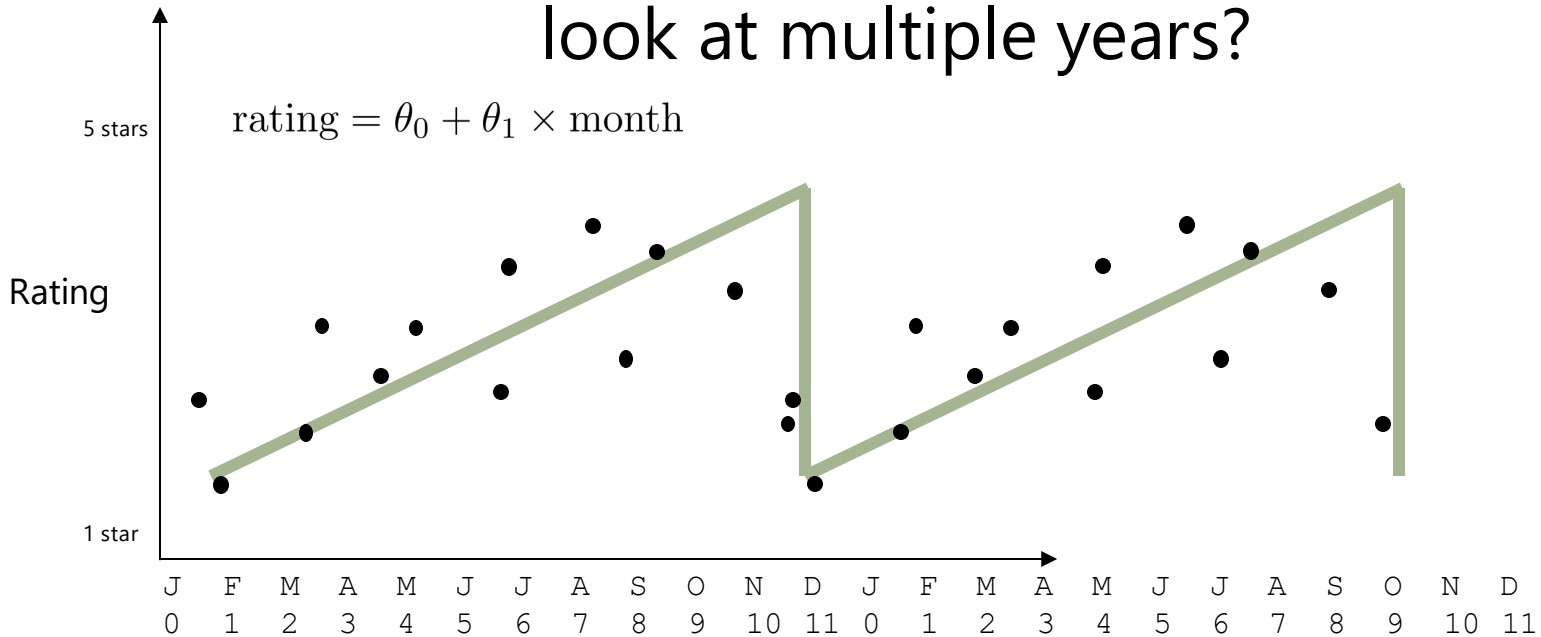
# Motivating examples

The model we'd learn might look something like:



# Motivating examples

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



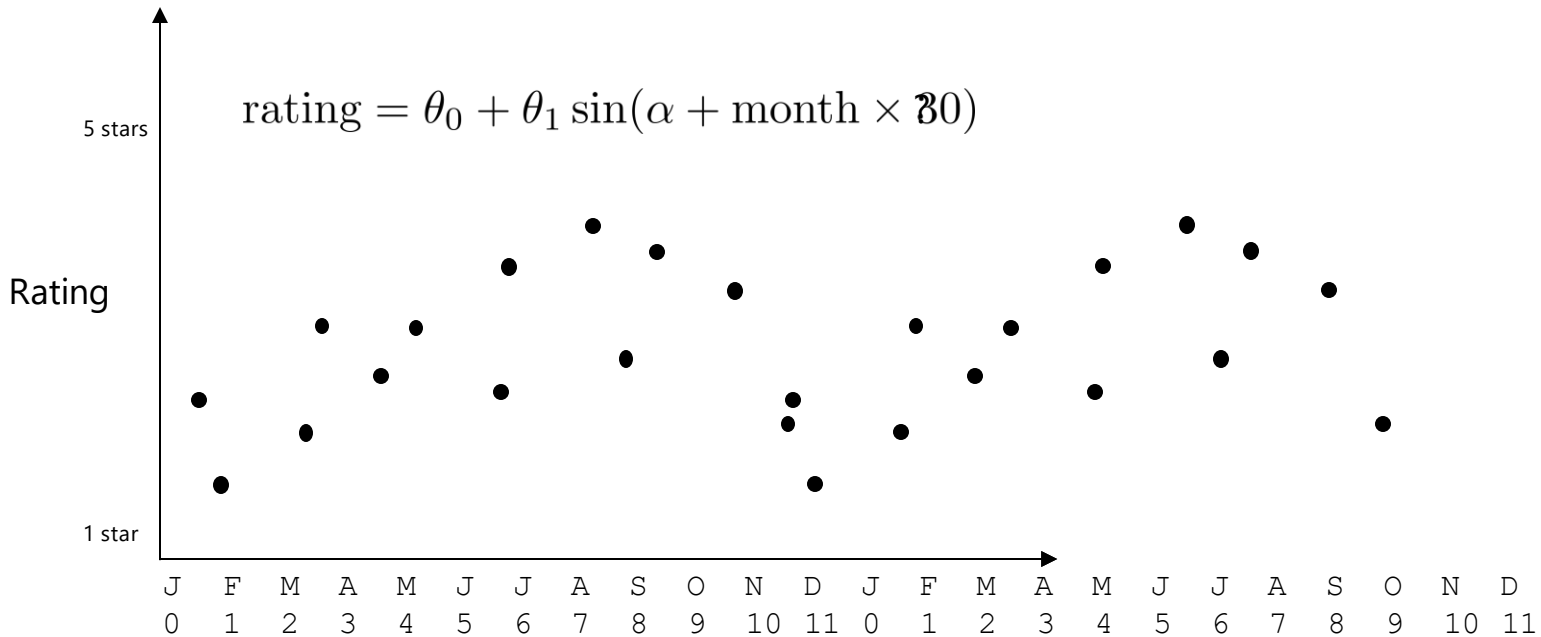
# Modeling temporal data

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 1<sup>st</sup> value.
- This type of “sawtooth” pattern probably isn’t very realistic

# Modeling temporal data

What might be a more realistic shape?



# Modeling temporal data

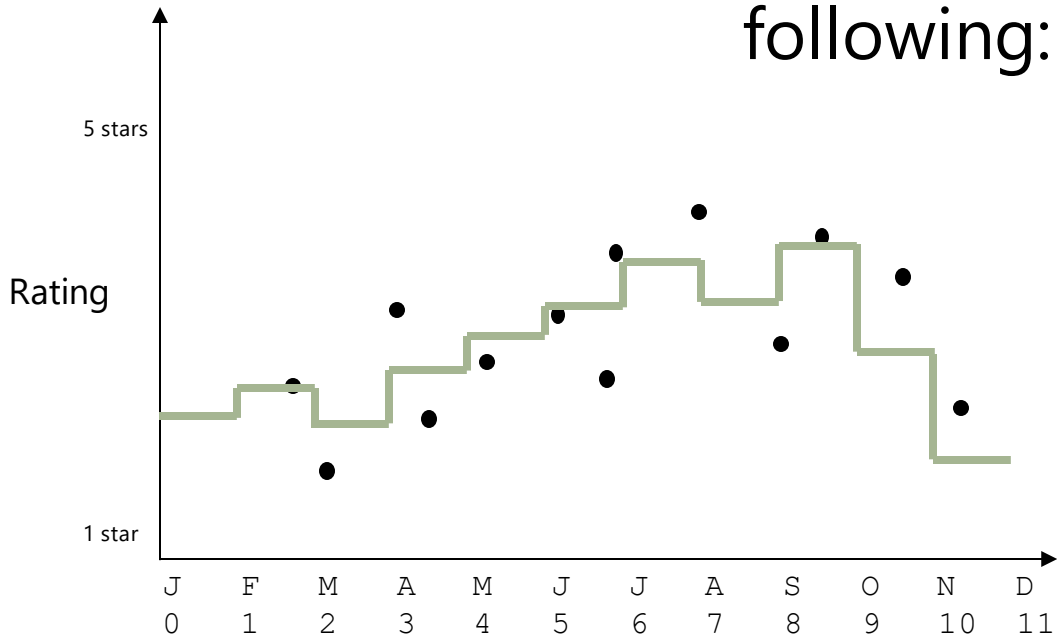
Fitting some periodic function like a sin wave would be a valid solution, but is difficult to get right, and fairly inflexible

- 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!



# Concept: Fitting piecewise functions

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



# Fitting piecewise functions

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.,  $\theta_0$  captures the January value,  $\theta_1$  captures the *difference* between February and January, etc.

# Fitting piecewise functions

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  
...  
 $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$  if December

# Fitting piecewise functions

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

# Concept: Combining one-hot encodings

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

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

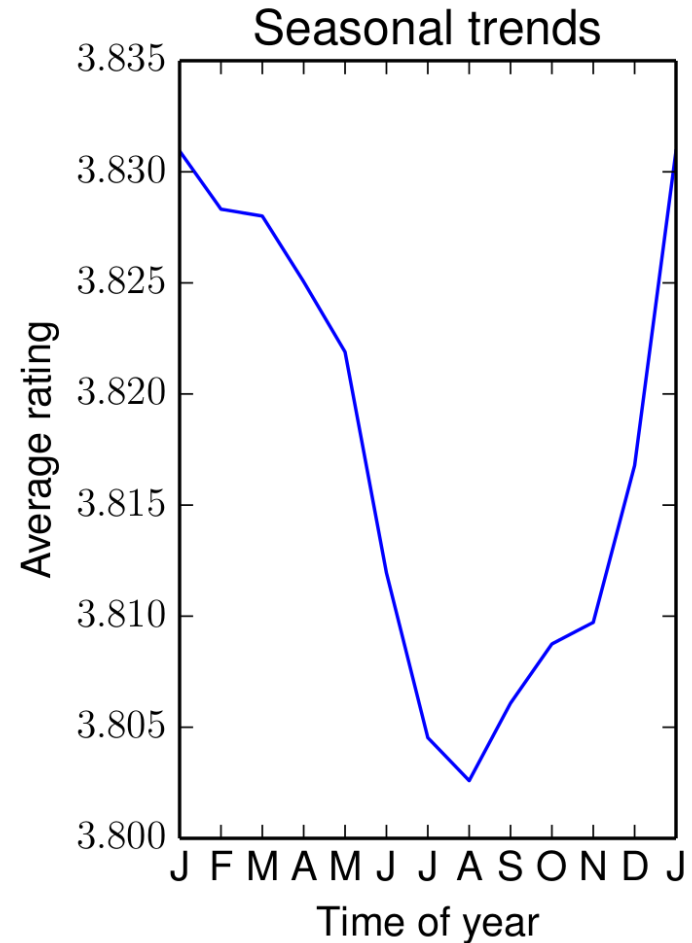
$x_1 = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$  if February  
 $[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$  if March  
 $[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$  if April  
...  
 $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$  if December

---

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

# What does the data actually look like?

Season vs.  
rating (overall)



# CSE 158 – Lecture 2

Web Mining and Recommender Systems

Regression Diagnostics

# Today: Regression diagnostics

## **Mean-squared error (MSE)**

$$\frac{1}{N} \|y - X\theta\|_2^2$$

$$= \frac{1}{N} \sum_{i=1}^N (y_i - X_i \cdot \theta)^2$$



# Regression diagnostics

**Q:** Why MSE (and not mean-absolute-error or something else)

# Regression diagnostics

# Regression diagnostics

## **Coefficient of determination**

**Q:** How low does the MSE have to be before it's "low enough"?

**A:** It depends! The MSE is proportional to the **variance** of the data

# Regression diagnostics

## **Coefficient of determination** ( $R^2$ statistic)

Mean:


Variance:

MSE:

## **Coefficient of determination** ( $R^2$ statistic)

$$FVU(f) = \frac{MSE(f)}{Var(y)}$$


(FVU = fraction of variance unexplained)

$FVU(f) = 1$   Trivial predictor

$FVU(f) = 0$   Perfect predictor

## **Coefficient of determination** ( $R^2$ statistic)

$$R^2 = 1 - FVU(f) = 1 - \frac{MSE(f)}{Var(y)}$$

$R^2 = 0$   Trivial predictor

$R^2 = 1$   Perfect predictor

# Overfitting

**Q:** But can't we get an  $R^2$  of 1 (MSE of 0) just by throwing in enough random features?

**A:** Yes! This is why MSE and  $R^2$  should always be evaluated on data that **wasn't** used to train the model

A good model is one that  
**generalizes to new data**



# Overfitting

When a model performs well on **training** data but doesn't generalize, we are said to be **overfitting**

# Overfitting

When a model performs well on **training** data but doesn't generalize, we are said to be **overfitting**

**Q:** What can be done to avoid overfitting?

# Occam's razor

"Among competing hypotheses, the one with the fewest assumptions should be selected"



# Occam's razor

$$X\theta = y$$

"hypothesis"



**Q:** What is a "complex" versus a "simple" hypothesis?

# Occam's razor

# Occam's razor

**A1:** A "simple" model is one where  $\theta$  has few non-zero parameters  
(only a few features are relevant)

**A2:** A "simple" model is one where  $\theta$  is almost uniform  
(few features are significantly more relevant than others)

# Occam's razor

**A1:** A "simple" model is one where theta has few non-zero parameters  $\longrightarrow \|\theta\|_1$  is small

**A2:** A "simple" model is one where theta is almost uniform  $\longrightarrow \|\theta\|_2$  is small

“Proof”



# Regularization

**Regularization** is the process of penalizing model complexity during training

$$\arg \min_{\theta} = \frac{1}{N} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2$$

MSE



(l2) model complexity



# Regularization

**Regularization** is the process of penalizing model complexity during training

$$\arg \min_{\theta} = \frac{1}{N} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2$$

How much should we trade-off accuracy versus complexity?



# Optimizing the (regularized) model

$$\arg \min_{\theta} = \underbrace{\frac{1}{N} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2}_{f(\theta)}$$

- Could look for a closed form solution as we did before
- Or, we can try to solve using **gradient descent**

# Optimizing the (regularized) model

## Gradient descent:

1. Initialize  $\theta$  at random
2. While (not converged) do  
$$\theta := \theta - \alpha f'(\theta)$$

All sorts of annoying issues:

- How to initialize theta?
- How to determine when the process has converged?
- How to set the step size alpha

These aren't really the point of this class though

# Optimizing the (regularized) model

$$f(\theta) = \frac{1}{N} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2$$

$$\frac{\partial f}{\partial \theta_k} \text{ ?}$$


# Optimizing the (regularized) model

## Gradient descent in scipy:

(code for all examples is on <http://jmcauley.ucsd.edu/cse158/code/week1.py>)

(see "ridge regression" in the "sklearn" module)

# Model selection

$$\arg \min_{\theta} = \frac{1}{N} \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2$$


How much should we trade-off accuracy versus complexity?

Each value of lambda generates a different model. **Q:** How do we select which one is the best?

# Model selection

How to select which model is best?

**A1:** The one with the lowest training error?

**A2:** The one with the lowest test error?

We need a **third** sample of the data that is not used for training or testing



# Model selection

A **validation set** is constructed to “tune” the model’s parameters

- Training set: used to **optimize the model’s parameters**
- Test set: used to report how well we expect the model to perform on **unseen data**
- Validation set: used to **tune** any model parameters that are not directly optimized

# Model selection

## A few “theorems” about training, validation, and test sets

- The training error **increases** as lambda **increases**
- The validation and test error are at least as large as the training error (assuming infinitely large random partitions)
- The validation/test error will usually have a “sweet spot” between under- and over-fitting

# Model selection

# Summary of Week 1: Regression

- Linear regression and least-squares
  - (a little bit of) feature design
  - Overfitting and regularization
    - Gradient descent
- Training, validation, and testing
  - Model selection

# Homework

Homework is **available** on the course  
webpage

[http://cseweb.ucsd.edu/classes/fa19/cse158-  
a/files/homework1.pdf](http://cseweb.ucsd.edu/classes/fa19/cse158-a/files/homework1.pdf)

Please submit it at the beginning of the  
**week 3** lecture (Oct 14)

All submissions should be made as **pdf**  
**files on gradescope**

Questions?