Web Mining and Recommender Systems

Supervised learning – Regression
Supervised learning is the process of trying to infer from labeled data the underlying function that produced the labels associated with the data.
What is supervised learning?

Given **labeled training data** of the form
\[
\{(\text{data}_1, \text{label}_1), \ldots, (\text{data}_n, \text{label}_n)\}
\]

Infer the function
\[ f(\text{data}) \rightarrow \text{labels} \]
Example

Suppose we want to build a movie recommender

e.g. which of these films will I rate highest?
Q: What are the labels?
A: ratings that others have given to each movie, and that I have given to other movies
Q: What is the data?

A: features about the movie and the users who evaluated it

Movie features: genre, actors, rating, length, etc.

User features: age, gender, location, etc.
Example

Movie recommendation:

\[ f(\text{data}) \overset{?}{\rightarrow} \text{labels} \]

\[ = \]

\[ f(\text{user features, movie features}) \overset{?}{\rightarrow} \text{star rating} \]
Solution 1

Design a system based on **prior knowledge**, e.g.

```python
def prediction(user, movie):
    if (user['age'] <= 14):
        if (movie['mpaa_rating']) == "G"):
            return 5.0
        else:
            return 1.0
    else if (user['age'] <= 18):
        if (movie['mpaa_rating']) == "PG"):
            return 5.0
    .... Etc.
```

Is this **supervised learning**?
Solution 2

Identify words that I frequently mention in my social media posts, and recommend movies whose plot synopses use similar types of language.

Is this **supervised learning**?

argmax similarity(synopsis, post)
Identify which attributes (e.g. actors, genres) are associated with positive ratings. Recommend movies that exhibit those attributes.

Is this **supervised learning**?
Solution 1

(design a system based on prior knowledge)

Disadvantages:
• Depends on possibly false assumptions about how users relate to items
• Cannot adapt to new data/information

Advantages:
• Requires no data!
Solution 2

(identify similarity between wall posts and synopses)

Disadvantages:
• Depends on possibly false assumptions about how users relate to items
• May not be adaptable to new settings

Advantages:
• Requires data, but does not require labeled data
Solution 3

(identify attributes that are associated with positive ratings)

Disadvantages:
- Requires a (possibly large) dataset of movies with labeled ratings

Advantages:
- Directly optimizes a measure we care about (predicting ratings)
- Easy to adapt to new settings and data
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 is one of the simplest supervised learning approaches to learn relationships between input variables (features) and output variables (predictions)
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)
Motivation: height vs. weight

Q: Can we find a line that (approximately) fits the data?

\[ y = mx + 6 \]

Weight \sim m \cdot \text{Height} + 6
**Q:** Can we find a line that (approximately) fits the data?

- If we can find such a line, we can use it to make **predictions** (i.e., estimate a person's weight given their height)
- How do we **formulate** the problem of finding a line?
- If no line will fit the data exactly, how to **approximate**?
  - What is the "best" line?
Recap: equation for a line

What is the formula describing the line?

\[ y = mx + b \]

Weight = \( m \times \) Height + \( b \)
Recap: equation for a line (plane)

What about in more dimensions?

Weight = \( m_1 \times \text{Height} + m_2 \times \text{age} + b \)

\[ y = m_1 x_1 + m_2 x_2 + b \]
Recap: equation for a line as an inner product

What about in more dimensions?

Weight = (Height, age, 1) \cdot (m_1, m_2, b)

y = (x_1, x_2, 1) \cdot (m_1, m_2, b)
**Linear regression** assumes a predictor of the form

\[ X \theta = y \]

**Q:** Solve for theta  
**A:**  
\[ \theta = (X^T X)^{-1} X^T y \]
**Linear regression** assumes a predictor of the form

\[ X\theta = y \]

**Q:** Solve for theta  
**A:** \[ \theta = (X^TX)^{-1}X^Ty \]
Web Mining and Recommender Systems

Worked Example – Regression
How do preferences toward certain beers vary with age?
Example 1

**Beer Advocate**

**Beers:**

**Ratings/reviews:**

- **4.35/5**
- **Overall:** 4.25
- **Look:** 4.5
- **Taste:** 4.5
- **Smell:** 4.25
- **Drinkability:** 4.5

* * *

**User profiles:**

- **HipCzech**
  - Affiliation: Ales Corner
  - **Male, from Texas**
  - **Profile Page**
    - **Member Since:** Jul 12, 2014
    - **Beers:** 108
    - **Pairs:** 6
    - **Trade:** Today at 12:15 AM
Example 1

50,000 reviews are available on
http://cseweb.ucsd.edu/classes/fa21/cse258-b/data/beer_50000.json
(see course webpage)
Example 1

Real-valued features

How do preferences toward certain beers vary with age?
How about ABV?

rating = \( O_0 + O_{1\text{ age}} \) + \( O_0 + O_{1\text{ ABV}} \)

(code for all examples is on the course webpage)
Example 1

Real-valued features

What is the interpretation of:

$$\theta = (3.4, 10e^{-7})$$

(code for all examples is on the course webpage)
Example 2

Categorical features

How do beer preferences vary as a function of gender?

(code for all examples is on the course webpage)
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!

$$X = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{male} \\ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{female}$$
How would you build a feature to represent the month, and the impact it has on people’s rating behavior?
Web Mining and Recommender Systems

Regression – Feature Transforms & Worked Example
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 is one of the simplest supervised learning approaches to learn relationships between input variables (features) and output variables (predictions).
Linear regression assumes a predictor of the form

\[ X\theta = y \]

(or \( Ax = b \) if you prefer)
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

Beers:

Ratings/reviews:

User profiles:
Real-valued features

How do preferences toward certain beers vary with age?
How about **ABV**?

(code for all examples on course webpage)
Example: 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:

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

- 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
Web Mining and Recommender Systems

Regression – Categorical Features
Categorical features

How do beer preferences vary as a function of gender?

(code for all examples is the course webpage)
E.g. How does rating vary with gender?
Example

\( \theta_0 \) 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”

- Rating = \theta_0 \quad \text{if male}
- Rating = \theta_0 + \theta_1 \quad \text{if female}
- Rating = \theta_0 + 2\theta_1 \quad \text{if other}
- Rating = \theta_0 + 3\theta_1 \quad \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:

Gender

Rating

male    female    other    not specified
Motivating examples

Instead we need something like:

\[
\text{Rating} = \theta_0 \quad \text{if male}
\]
\[
\text{Rating} = \theta_0 + \theta_1 \quad \text{if female}
\]
\[
\text{Rating} = \theta_0 + \theta_2 \quad \text{if other}
\]
\[
\text{Rating} = \theta_0 + \theta_3 \quad \text{if not specified}
\]
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

- 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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature = [1, 0, 0]</td>
<td>for “female”</td>
</tr>
<tr>
<td>Feature = [0, 1, 0]</td>
<td>for “other”</td>
</tr>
<tr>
<td>Feature = [0, 0, 1]</td>
<td>for “not specified”</td>
</tr>
</tbody>
</table>
Linearity dependent features

\[\text{male} = \begin{bmatrix} 1 \\ 10 \end{bmatrix} \]

\[\text{female} = \begin{bmatrix} 1 \\ 01 \end{bmatrix} \]

\[X = \begin{bmatrix} 1 & 1 & 6 \\ 1 & 0 & 10 \\ 1 & 1 & 01 \\ 1 & 0 & 1 \end{bmatrix} \]

\[X^T X = \begin{bmatrix} 5 & 2 & 3 \\ 2 & 2 & 0 \\ 3 & 0 & 3 \end{bmatrix} \begin{bmatrix} a + b \\ b \\ a \end{bmatrix}\]
Linearly dependent features

\[ \text{rating} = \theta_0 + \theta_1 M + \theta_2 F \]
\[ = 4 + 0.5M + 0.2F \]
\[ = 1000 - 995.57 - 995.8F \]

\( \infty \) identical solutions
Web Mining and Recommender Systems

Regression – Temporal Features
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?
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:

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

\[
\begin{align*}
\text{Jan} &= [0] \\
\text{Feb} &= [1] \\
\text{Mar} &= [2] \\
\text{etc.}
\end{align*}
\]
Motivating examples

The model we’d learn might look something like:

\[ \text{rating} = \theta_0 + \theta_1 \times \text{month} \]
Motivating examples

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

\[ \text{rating} = \theta_0 + \theta_1 \times \text{month} \]
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?

rating = $\theta_0 + \theta_1 \sin(\alpha + \text{month} \times 30)$?
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!
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:

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

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:

\[ \text{rating} = \theta \cdot x \quad \text{where} \]

\[ x = [1,1,0,0,0,0,0,0,0,0,0,0] \text{ if February} \]
\[ [1,0,1,0,0,0,0,0,0,0,0,0] \text{ if March} \]
\[ [1,0,0,1,0,0,0,0,0,0,0,0] \text{ if April} \]
\[ \ldots \]
\[ [1,0,0,0,0,0,0,0,0,0,0,1] \text{ if December} \]
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.
We can also extend this by combining several one-hot encodings together:

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

\[
x_1 = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if February}
\]
\[
[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if March}
\]
\[
[1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if April}
\]
\[
\ldots
\]
\[
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0] \quad \text{if December}
\]

\[
x_2 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if Tuesday}
\]
\[
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if Wednesday}
\]
\[
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] \quad \text{if Thursday}
\]
\[
\ldots
\]
What does the data actually look like?

Season vs. rating (overall)
Web Mining and Recommender Systems

Regression Diagnostics
Today: Regression diagnostics

**Mean-squared error (MSE)**

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

\[= \frac{1}{N} \sum_{i=1}^{N} (y_i - X_i \cdot \theta)^2 \]
Q: Why MSE (and not mean-absolute-error or something else)
Regression diagnostics

\[ \hat{y}_i = y_i - x_i \beta \]

\[ \varepsilon \]

\[ \varepsilon / |d| \leq 0.5 \]

\[ |e_i| > 0.5 \]
Regression diagnostics

\[ d = y_i - x_i \cdot \theta \]

\[ y_i = x_i \cdot \theta + N(0, \sigma^2) \]

\[ p_{\theta}(y|x) = \prod_{i} \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(y_i - x_i \cdot \theta)^2}{\sigma^2}} \]

\[ \max_{\theta} p_{\theta}(y|x) = \max_{\theta} \prod_{i} e^{-\frac{(y_i - x_i \cdot \theta)^2}{\sigma^2}} = \min_{\theta} \sum_{i} (y_i - x_i \cdot \theta)^2 \]
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.
Coefficient of determination
(R^2 statistic)

Mean: \( \bar{y} = \frac{1}{N} \sum y_i \)

Variance: \( \text{var}(y) = \frac{1}{N} \sum (y_i - \bar{y})^2 \)

MSE: \( = \frac{1}{n} \sum (y_i - x_i \theta)^2 \)
Regression diagnostics

Coefficient of determination
(R^2 statistic)

Mean: \( \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \)

Variance: \( \text{Var}(y) = \frac{1}{N} \sum_{i=1}^{N} (\bar{y} - y_i)^2 \)

MSE: \( \frac{1}{N} \sum_{i=1}^{N} (X_i \cdot \theta - y_i)^2 \)
Regression diagnostics

Coefficient of determination
(R^2 statistic)

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

(FVU = fraction of variance unexplained)

\[ FVU(f) = 1 \quad \text{Trivial predictor} \]
\[ FVU(f) = 0 \quad \text{Perfect predictor} \]
Regression diagnostics

Coefficient of determination
(R^2 statistic)

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

\[ R^2 = 0 \quad \rightarrow \quad \text{Trivial predictor} \]
\[ R^2 = 1 \quad \rightarrow \quad \text{Perfect predictor} \]
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.
When a model performs well on \textit{training} data but doesn’t generalize, we are said to be \textit{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”
Q: What is a “complex” versus a “simple” hypothesis?
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

\[ \| \theta \|_1 \text{ is small} \]

**A2:** A “simple” model is one where theta is almost uniform

\[ \| \theta \|_2 \text{ is small} \]
“Proof”
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 is the process of penalizing model complexity during training

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

How much should we trade-off accuracy versus complexity?
Optimizing the (regularized) model

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

- 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

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

\[
\frac{\partial f}{\partial \theta_k}
\]
Optimizing the (regularized) model

Gradient descent in scipy: code on course webpage

(see also “ridge regression” in the “sklearn” module)
Web Mining and Recommender Systems

Model Selection & Summary
Model selection

\[ \text{arg min}_\theta = \frac{1}{N} \| y - X \theta \|^2_2 + \lambda \| \theta \|^2_2 \]

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?
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
Homework is available on the course webpage
http://cseweb.ucsd.edu/classes/fa21/cse258-b/files/homework1.pdf

Please submit it by the beginning of the week 3 lecture

All submissions should be made as pdf files on gradescope