Lecture 13 – Feature Engineering, Clustering

DSC 40A, Winter 2024

Midterm 1 is Friday during lecture

- Formula sheet will be provided for you. No other notes.
- No calculators. This implies no crazy calculations.
- Assigned seats will be posted on Course Website and Campuswire.
- We will not answer questions during the exam. State your assumptions if anything is unclear.
- The exam will include long-answer homework-style questions, as well as short-answer questions such as True/False or filling in a numerical answer.
- The exam covers Lecture 1 to Lecture 12 (HW1-3 + Linear Algebra and Multiple Linear Regression).

Midterm study strategy

- Look at annotated lecture notes.
- Review the written solutions to previous homeworks and groupworks.
- Identify which concepts are still uncertain. Re-watch podcasts, post on Campuswire, come to office hours, use resources on course website, watch Janine's lecture videos.
- Work through past exams on course website and the posted mock exam.
- Study in groups.
- Summarize key facts and formulas.

Some Tips About Midterm

- Understand the derivations we did in lecture.
  - There will be derivation problems in the midterm, but no long derivation.
  - Understand the derivation I did in lecture and some methods I used
- Be able to perform simple algebra, calculus and linear algebra computation
  - Example: calculating matrix multiplication.
- Read each question carefully.
  - Example: using formal definition to prove convexity vs. using any method to prove convexity.
- Some problems are easier, some problems are harder.

Extra Credit Opportunity

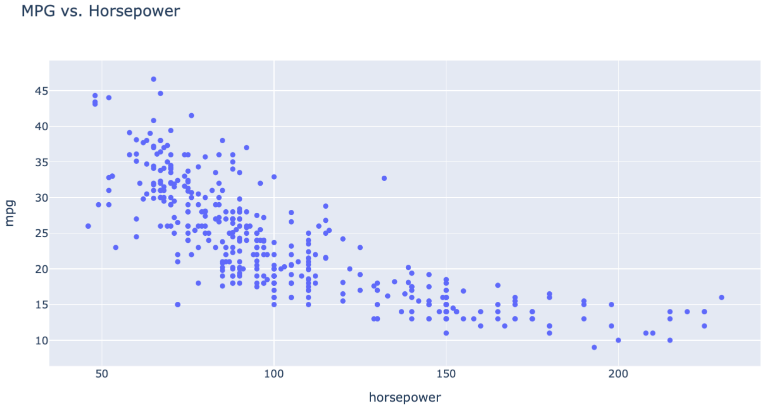
- The last problem on HW4 will be a class-wide competition on finding energies for High-Purity Germanium Detector waveforms
  - I'll explain what this is on next Monday's lecture, after the exam.
- Top predictions will get extra credit on Midterm 1.
- More detail next Monday

Agenda

- Feature engineering.
- Taxonomy of machine learning.
- Clustering.

Feature engineering

Last time: Cars



Question: Would a linear prediction rule work well on this dataset?

A quadratic prediction rule

- It looks like there's some sort of quadratic relationship between horsepower and MPG in the last scatter plot. We want to try and fit a prediction rule of the form

$$H(x) = w_0 + w_1 x + w_2 x^2$$

  - Note that while this is quadratic in horsepower, it is **linear in the parameters**!

- We can do that, by choosing our two "features" to be x_i and x_i^2, respectively.
  - In other words, $x_i^{(1)} = x_i$ and $x_i^{(2)} = x_i^2$.

  - More generally, we can create new features out of existing features.

A quadratic prediction rule

- Desired prediction rule: $H(x) = w_0 + w_1 x + w_2 x^2$.
- The resulting design matrix looks like this:

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \dots & & \\ 1 & x_n & x_n^2 \end{bmatrix}$$

- To find optimal parameter vector $\vec{w}^*$: solve the **normal equations**!

$$X^T X w^* = X^T y$$

More examples

- What if we want to use a prediction rule of the form $H(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$?

- What if we want to use a prediction rule of the form $H(x) = w_1 \frac{1}{x^2} + w_2 \sin x + w_3 e^x$?

Feature engineering

- The process of creating new features out of existing information in our dataset is called **feature engineering**.
  - In this class, feature engineering will mostly be restricted to creating non-linear functions of existing features (as in the previous example).

  - In the future you'll learn how to do other things, like encode categorical information.

Non-linear functions of multiple features

- Recall our example from last lecture of predicting sales from square footage and number of competitors. What if we want a prediction rule of the form

$$
\begin{aligned}
H(\text{sqft}, \text{comp}) &= w_0 + w_1 \text{sqft} + w_2 \text{sqft}^2 \\
&\quad + w_3 \text{comp} + w_4 \text{sqft} \cdot \text{comp} \\
&= w_0 + w_1 s + w_2 s^2 + w_3 c + w_4 sc
\end{aligned}
$$

- Make design matrix:

$$
X = \begin{bmatrix} 1 & s_1 & s_1^2 & c_1 & s_1 c_1 \\ 1 & s_2 & s_2^2 & c_2 & s_2 c_2 \\ \dots & \dots & \dots & \dots & \\ 1 & s_n & s_n^2 & c_n & s_n c_n \end{bmatrix}
$$

Where s_i and c_i are square footage and number of competitors for store i, respectively.

Finding the optimal parameter vector, $\vec{w}^*$

- As long as the form of the prediction rule permits us to write $\vec{h} = X\vec{w}$ for some X and $\vec{w}$, the mean squared error is

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2$$

- Regardless of the values of X and $\vec{w}$,

$$\begin{aligned} &\frac{dR_{\text{sq}}}{d\vec{w}} = 0 \\ \implies &-2X^T\vec{y} + 2X^TX\vec{w} = 0 \\ \implies &X^TX\vec{w}^* = X^T\vec{y}. \end{aligned}$$

- The **normal equations** still hold true!

Linear in the parameters

- We can fit rules like:

$$w_0 + w_1 x + w_2 x^2 \qquad w_1 e^{-x^{(1)^2}} + w_2 \cos(x^{(2)} + \pi) + w_3 \frac{\log 2x^{(3)}}{x^{(2)}}$$

- This includes arbitrary polynomials.

- We can't fit rules like:

$$w_0 + e^{w_1 x} \qquad w_0 + \sin(w_1 x^{(1)} + w_2 x^{(2)})$$

- We can have any number of parameters, as long as our prediction rule is **linear in the parameters**, or linear when we think of it as a function of the parameters.

Determining function form

- How do we know what form our prediction rule should take?

- Sometimes, we know from *theory*, using knowledge about what the variables represent and how they should be related.

- Other times, we make a guess based on the data.

- Generally, start with simpler functions first.
  - Remember, the goal is to find a prediction rule that will generalize well to unseen data.

Discussion Question

Suppose you collect data on the height, or position, of a freefalling object at various times t_i. Which form should your prediction rule take to best fit the data?

a) constant, $H(t) = w_0$
b) linear, $H(t) = w_0 + w_1 t$
c) quadratic, $H(t) = w_0 + w_1 t + w_2 t^2$
d) no way to know without plotting the data

Example: Amdahl's Law

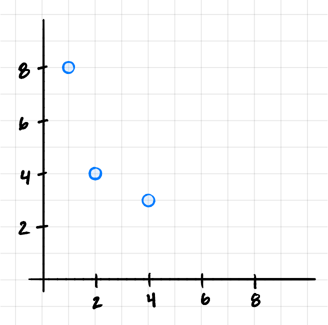
- Amdahl's Law relates the runtime of a program on p processors to the time to do the sequential and nonsequential parts on one processor.

$$H(p) = t_{\mathrm{S}} + \frac{t_{\mathrm{NS}}}{p}$$

- Collect data by timing a program with varying numbers of processors:

Processors	Time (Hours)
1	8
2	4
4	3

Example: fitting $H(x) = w_0 + w_1 \cdot \frac{1}{x}$



x_i	y_i
1	8
2	4
4	3

Example: Amdahl's Law

- The solution is: $t_S = 1, \quad t_{NS} = \frac{48}{7} \approx 6.86$
- Therefore our prediction rule is:

$$H(p) = t_S + \frac{t_{NS}}{p}$$

$$= 1 + \frac{6.86}{p}$$

Transformations

How do we fit prediction rules that aren't linear in the parameters?

- Suppose we want to fit the prediction rule

$$H(x) = w_0 e^{w_1 x}$$

  This is **not** linear in terms of w_0 and w_1, so our results for linear regression don't apply.

- **Possible Solution:** Try to apply a **transformation**.

Transformations

- **Question:** Can we re-write $H(x) = w_0 e^{w_1 x}$ as a prediction rule that **is** linear in the parameters?

Transformations

- **Solution:** Create a new prediction rule, $T(x)$, with parameters b_0 and b_1, where $T(x) = b_0 + b_1 x$.
  - This prediction rule is related to $H(x)$ by the relationship $T(x) = \log H(x)$.
  - $\vec{b}$ is related to $\vec{w}$ by $b_0 = \log w_0$ and $b_1 = w_1$.
  - Our new observation vector, $\vec{z}$, is $\begin{bmatrix} \log y_1 \\ \log y_2 \\ \dots \\ \log y_n \end{bmatrix}$.
- $T(x) = b_0 + b_1 x$ is linear in its parameters, b_0 and b_1.
- Use the solution to the normal equations to find $\vec{b}^*$, and the relationship between $\vec{b}$ and $\vec{w}$ to find $\vec{w}^*$.

Demo

Let’s try this out in a Jupyter notebook. Follow along here.

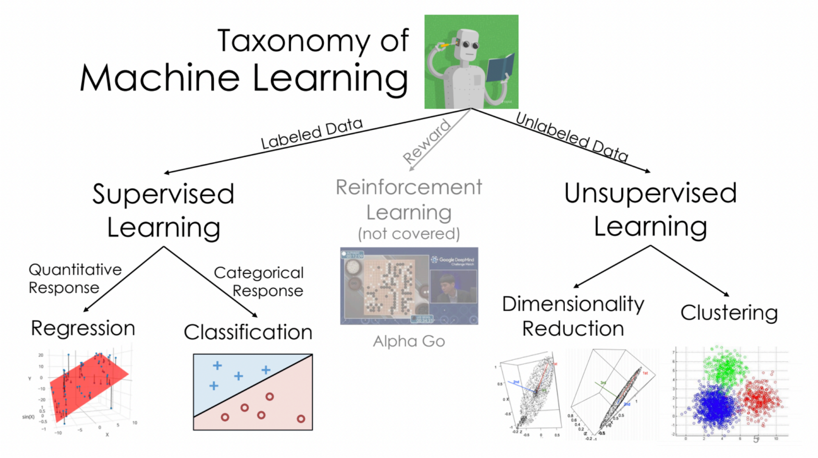
Non-linear prediction rules in general

- Sometimes, it's just not possible to transform a prediction rule to be linear in terms of some parameters.
- In those cases, you'd have to resort to other methods of finding the optimal parameters.
  - For example, with $H(x) = w_0 e^{w_1 x}$, we could use gradient descent or a similar method to minimize mean squared error, $R(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - w_0 e^{w_1 x_i}\right)^2$, and find w_0^*, w_1^* that way.
- Prediction rules that are linear in the parameters are much easier to work with.

Taxonomy of machine learning

What is machine learning?

- **One definition:** Machine learning is about getting a computer to find patterns in data.
- Have we been doing machine learning in this class? **Yes.**
  - Given a dataset containing salaries, predict what my future salary is going to be.
  - Given a dataset containing years of experience, GPAs, and salaries, predict what my future salary is going to be given my years of experience and GPA.

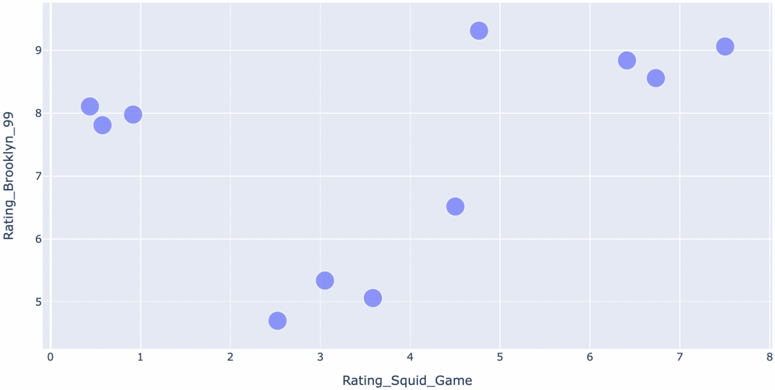


[1]

[1]taken from Joseph Gonzalez at UC Berkeley

Clustering

Question: how might we "cluster" these points into groups?



Problem statement: clustering

Goal: Given a list of n data points, stored as vectors in $\mathbb{R}^d$, $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$, and a positive integer k, **place the data points into** k **groups of nearby points**.

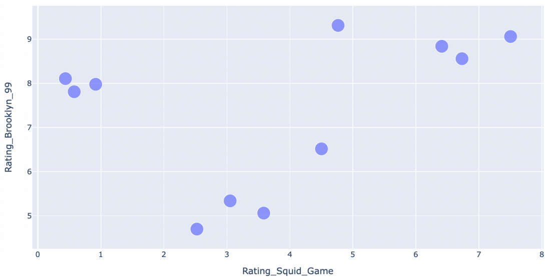
- These groups are called "clusters".
- Think about groups as **colors**.
  - i.e., the goal of clustering is to assign each point a color, such that points of the same color are close to one another.
- Note, unlike with regression, there is no "right answer" that we are trying to predict — there is no y!
  - Clustering is an **unsupervised** method.

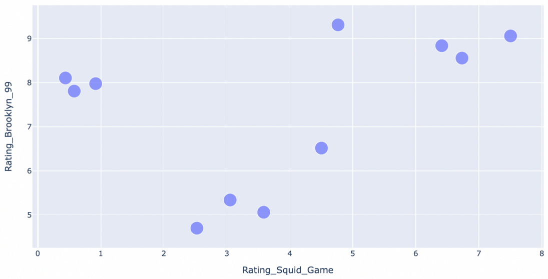
How do we define a group?

- One solution: pick k cluster centers, i.e. **centroids**:

$$\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_k \text{ in } \mathbb{R}^d$$

- These k centroids define the k groups.
- Each data point “belongs” to the group corresponding to the nearest centroid.
- This reduces our problem from being “find the best group for each data point” to being “find the best locations for the centroids”.


Rating_Brooklyn_99
Rating_Squid_Game
0 1 2 3 4 5 6 7 8
5 6 7 8 9


Rating_Brooklyn_99
Rating_Squid_Game
0 1 2 3 4 5 6 7 8
5 6 7 8 9

How do we pick the centroids?

- Let's come up with an **cost function**, C, which describes how good a set of centroids is.
  - Cost functions are a generalization of empirical risk functions.

- One possible cost function:

$$C(\mu_1, \mu_2, ..., \mu_k) = \text{total squared distance of each data point } \vec{x}_i \text{ to its closest centroid } \mu_j$$

- This C has a special name, **inertia**.

- Lower values of C lead to "better" clusterings.
  - **Goal:** Find the centroids $\mu_1, \mu_2, ..., \mu_k$ that minimize C.

Discussion Question

Suppose we have n data points, $\vec{x}_1, \vec{x}_2, ..., \vec{x}_n$, each of which are in $\mathbb{R}^d$.

Suppose we want to cluster our dataset into k clusters.

How many ways can we assign points to clusters?

a) $d \cdot k$
b) d^k
c) n^k
d) k^n
e) $n \cdot k \cdot d$

How do we minimize inertia?

- **Problem**: there are exponentially many possible clusterings. It would take too long to try them all.
- **Another Problem**: we can't use calculus or algebra to minimize C, since to calculate C we need to know which points are in which clusters.
- We need another solution.

k-Means Clustering, i.e. Lloyd's Algorithm

Here's an algorithm that attempts to minimize inertia:

1. Pick a value of k and randomly initialize k centroids.
2. Keep the centroids fixed, and update the groups.
   - Assign each point to the nearest centroid.
3. Keep the groups fixed, and update the centroids.
   - Move each centroid to the center of its group.
4. Repeat steps 2 and 3 until the centroids stop changing.

Example

See the following site for an interactive visualization of k-Means Clustering: `https://tinyurl.com/40akmeans`

Summary, next time

Summary

- The process of creating new features is called feature engineering.
- As long as our prediction rule is linear in terms of its parameters $w_0, w_1, ..., w_d$, we can use the solution to the normal equations to find $\vec{w}^*$.
    - Sometimes it's possible to transform a prediction rule into one that is linear in its parameters.
- Linear regression is a form of supervised machine learning, while clustering is a form of unsupervised learning.
- Clustering aims to place data points into "groups" of points that are close to one another. k-means clustering is one method for finding clusters.

Next time

- How does k-means clustering attempt to minimize inertia?
- How do we choose good initial centroids?
- How do we choose the value of k, the number of clusters?