Lecture 10 – Feature Engineering, Clustering

DSC 40A, Fall 2021 @ UC San Diego

Suraj Rampure, with help from **many others**

Announcements

- Groupwork 4 due **Thursday at 11:59pm**.

- Homework 4 due **Tuesday(!) at 11:59pm**.
    - Remember, everyone has 5 slip days.

- Survey 4 will come out this weekend, and is also due **Tuesday at 11:59pm**.

- The office hours schedule has changed! Look at the calendar.

Agenda

- Feature engineering.

- Taxonomy of machine learning.

- Clustering.

Feature engineering

The general problem

- We have n data points (or **training examples**): $(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)$ where each $\vec{x}_i$ is a feature vector of d features:

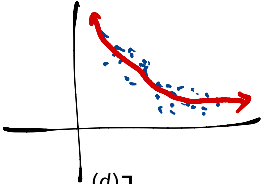
$$\vec{x}_i = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \dots \\ x_i^{(d)} \end{bmatrix} \begin{matrix} \to \text{experience}_i \\ \to \text{GPA}_i \\ \vdots \\ \ \end{matrix}$$

- We want to find a good linear prediction rule:

$$\begin{aligned} H(\vec{x}) &= w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \dots + w_d x^{(d)} \\ &= \vec{w} \cdot \text{Aug}(\vec{x}) \end{aligned}$$

$$\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \qquad \text{Aug}(\vec{x}) = \begin{bmatrix} 1 \\ x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(d)} \end{bmatrix}$$

The general solution



- Use design matrix

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(d)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(d)} \\ \dots & \dots & \dots & & \dots \\ 1 & x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(d)} \end{bmatrix} = \begin{bmatrix} \mathrm{Aug}(\vec{x}_1)^T \\ \mathrm{Aug}(\vec{x}_2)^T \\ \dots \\ \mathrm{Aug}(\vec{x}_n)^T \end{bmatrix}$$

  and observation vector to solve the **normal equations**

$$X^T X \vec{w}^* = X^T \vec{y}$$

  to find the optimal parameter vector $\vec{w}^*$.

- **Feature engineering**: creating new features out of existing features in order to better fit the data.

Example

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

$$X = \begin{bmatrix} \frac{1}{x_1^2} & \sin x_1 & e^{x_1} \\ \frac{1}{x_2^2} & \sin x_2 & e^{x_2} \\ \vdots & \vdots & \vdots \\ \frac{1}{x_n^2} & \sin x_n & e^{x_n} \end{bmatrix} \qquad \vec{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$\vec{h} = X\vec{w}$$

$H(x_1) =$ first row of $X\vec{w}$

$$= w_1 \cdot \frac{1}{x_1^2} + w_2 \cdot \sin x_1 + w_3 \cdot e^{x_1}$$

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:

$$
\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}
\qquad
X = \begin{bmatrix} 1 & s_1 & s_1^2 & c_1 & s_1 c_1 \\ 1 & s_2 & s_2^2 & c_2 & s_2 c_2 \\ \dots & \dots & \dots & \dots & \\ 1 & s_n & s_n^2 & c_n & s_n c_n \end{bmatrix}
$$

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

$$\vec{h} = X\vec{w}$$

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 $\vec{w}$,

$$\frac{dR_{\text{sq}}}{d\vec{w}} = 0$$
$$\implies -2X^T\vec{y} + 2X^TX\vec{w} = 0$$
$$\implies X^TX\vec{w}^* = X^T\vec{y}. \quad \rightsquigarrow \vec{w}^* = (X^TX)^{-1}X^T\vec{y}$$

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

Linear in the parameters

- We can fit rules like:

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

- This includes arbitrary polynomials.

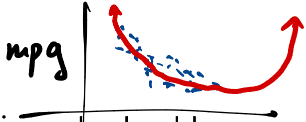
- We can't fit rules like:

$X\vec{w}$

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

- We can have any number of parameters, as long as our prediction rule is **linear in the parameters**.

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.
  - See Homework 4, Question 2D and 2E.

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?

$A(t) = 9.81$

$V(t) = 9.81t + C_0$

$D(t) = \frac{9.81}{2}t^2 + C_0 t + C_1$

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~~

To answer, go to `menti.com` **and enter 2657 7681.**

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.

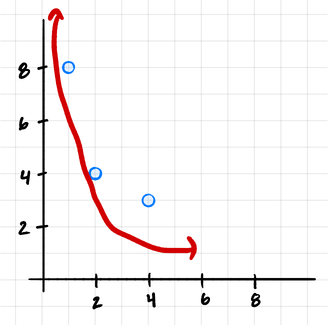
$$H(x) = w_0 + w_1 \cdot \frac{1}{x}$$

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

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

x_i Processors	y_i 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

$$X = \begin{bmatrix} 1 & 1 \\ 1 & \frac{1}{2} \\ 1 & \frac{1}{4} \end{bmatrix}$$

$$\vec{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \qquad \vec{y} = \begin{bmatrix} 8 \\ 4 \\ 3 \end{bmatrix}$$

$$X^T X \vec{w}^* = X^T \vec{y}$$

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

vector with w_0^*, w_1^*

Example: Amdahl's Law

- We found: $t_S = 1, \quad t_{NS} = \frac{48}{7} \approx 6.86$

- 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 $X\vec{w}$

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

  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

$\log(ab)$
$= \log(a) + \log(b)$

Hint: log!

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

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

$$\log H(x) = \log\left(w_0 e^{w_1 x}\right)$$

$$= \log w_0 + \log e^{w_1 x}$$

$$= \log w_0 + w_1 x \log e$$

$$\underbrace{\log H(x)}_{} = \underbrace{\log w_0}_{b_0} + \underbrace{w_1}_{b_1} x$$

$$T(x) = b_0 + b_1 x$$

Transformations

$$\underbrace{\log H(x)}_{T(x)} = \underbrace{\log w_0}_{b_0} + \underbrace{w_1}_{b_1} x$$

- **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 $\underbrace{\begin{bmatrix} \log y_1 \\ \log y_2 \\ \dots \\ \log y_n \end{bmatrix}}$.

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \qquad X^T X \vec{b}^* = X^T \vec{z}$$

- $T(x) = b_0 + b_1 x$ 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 $\vec{w}^*$.

Follow along with the demo by clicking the **code** link on the course website next to Lecture 10.

$$H(x) = w_0 + e^{w_1 x} + w_2 \sin(w_3 x)$$

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \left(w_0 + e^{w_1 x_i} + w_2 \sin(w_3 x_i)\right)\right)^2$$

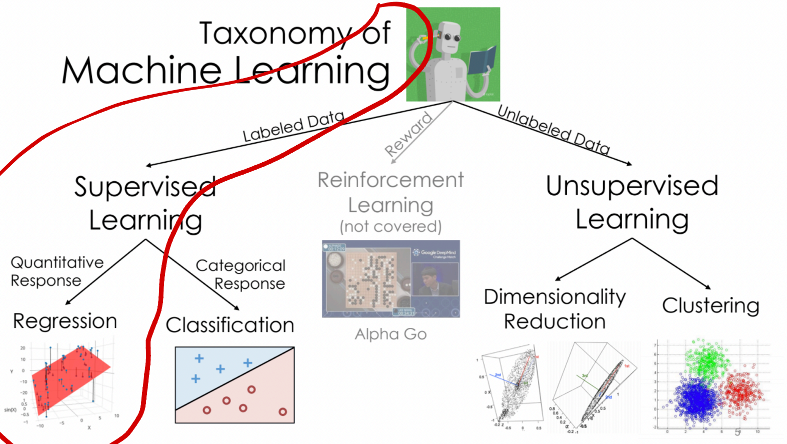
Non-linear prediction rules in general

- 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.

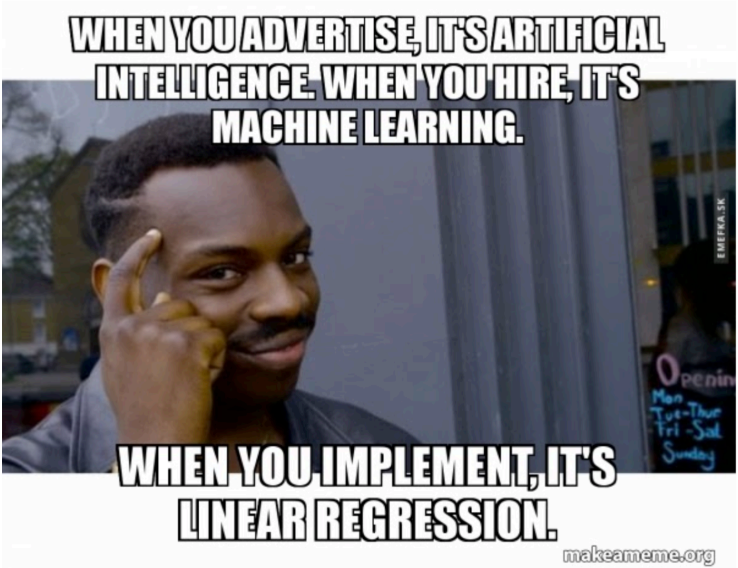


so far: we have the "right answers" in our dataset

e.g. y = right answers

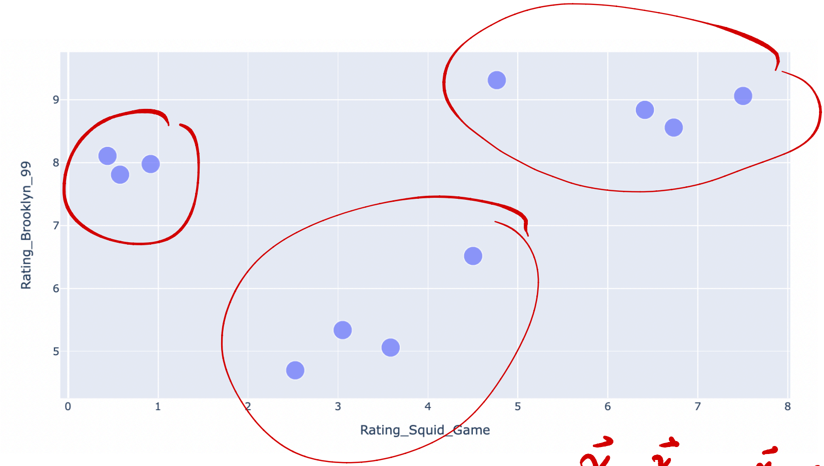
[1]

[1]taken from Joseph Gonzalez @ UC Berkeley


WHEN YOU ADVERTISE, IT'S ARTIFICIAL INTELLIGENCE. WHEN YOU HIRE, IT'S MACHINE LEARNING.
EMEFKA.SK
WHEN YOU IMPLEMENT, IT'S LINEAR REGRESSION.
makeameme.org

Clustering

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



$\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^2$

Problem statement: clustering

Goal: Given a list of n data points, stored as vectors in $\mathbb{R}^d$, $\vec{x}_1, \vec{x}_2, ..., \vec{x}_n$, 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?

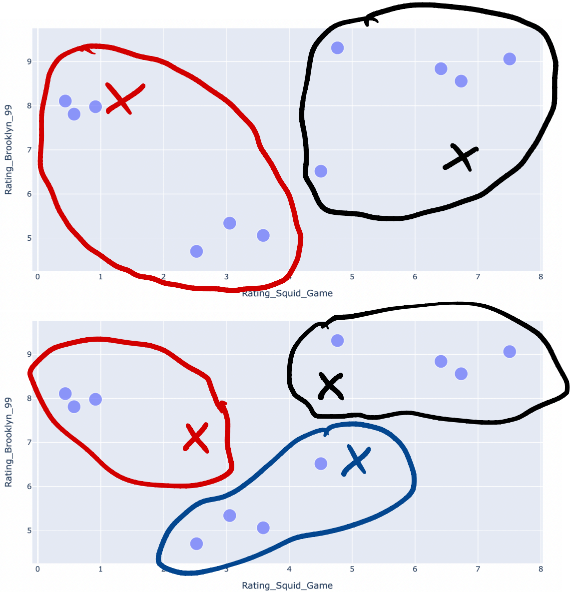
- One solution: pick k cluster centers, i.e. **centroids**:

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

- 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
Rating_Brooklyn_99
Rating_Squid_Game

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:

absolute?
cosine?

$C(\mu_1, \mu_2, ..., \mu_k) =$ total squared distance of each data point $\vec{x}_i$ to its closest centroid μ_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 I assign points to clusters?

A) ~~$d \cdot k$~~
B) ~~d^k~~
C) n^k
D) k^n
E) ~~$n \cdot k \cdot d$~~

$$\underset{\vec{x}_1}{\underline{k}} \cdot \underset{\vec{x}_2}{\underline{k}} \cdot \ldots \cdot \underset{\vec{x}_n}{\underline{k}} = k^n$$

To answer, go to `menti.com` and enter 2657 7681.

How do we minimize inertia?

K^n

- **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.

"NP Hard"

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

Here's an algorithm that attemps 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 $w_0, w_1, \ldots, w_d$, 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?