Lecture 13 – Feature Engineering

DSC 40A, Fall 2022 @ UC San Diego

Mahdi Soleymani, with help from **many others**

Agenda

- Interpreting weights.
- Feature engineering.
- Taxonomy of machine learning.

Which features are most "important"?

- The most important feature is **not necessarily** the feature with largest weight.

- Features are measured in different units, scales.
  - Suppose I fit one prediction rule, H_1, with sales in dollars, and another prediction rule, H_2, with sales in thousands of dollars.
  - Sales is just as important in both prediction rules.
  - But the weight of sales in H_1 will be 1000 times smaller than the weight of sales in H_2.
  - Intuitive explanation: $5 \times 45000 = (5 \times 1000) \times 45$.

- **Solution**: we should **standardize** each feature, i.e. convert each feature to standard units.

Standard units

- Recall from Lecture 6: to convert a feature $x_1, x_2, \ldots, x_n$ to standard units, we use the formula

$$x_i \text{ in standard units} = \frac{x_i - \bar{x}}{\sigma_x}$$

- Example: 1, 7, 7, 9
  - Mean: 6
  - Standard deviation:

$$\sqrt{\frac{1}{4}((-5)^2 + (1)^2 + (1)^2 + (3)^2)} = 3$$

  - Standardized data:

$$\frac{1-6}{3} = -\frac{5}{3}, \qquad \frac{7-6}{3} = \frac{1}{3}, \qquad \frac{7-6}{3} = \frac{1}{3}, \qquad \frac{9-6}{3} = 1$$

Standard units for multiple linear regression

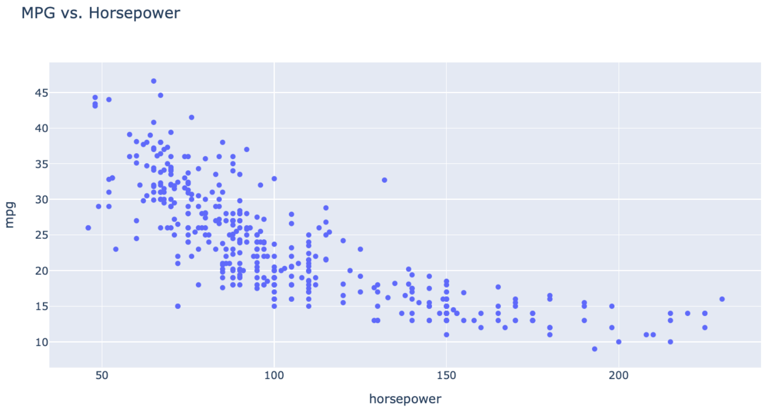
- The result of standardizing each feature (separately!) is that the units of each feature are on the same scale.
  - There's no need to standardize the outcome (net sales), since it's not being compared to anything.

- Then, solve the normal equations. The resulting $w_0^*, w_1^*, \ldots, w_d^*$ are called the **standardized regression coefficients**.

- Standardized regression coefficients can be directly compared to one another.

Let's jump back to our demo notebook.

Feature engineering



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 this still a linear model, because 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

- More generally, we can create new features out of existing information in our dataset. This process 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.

Feature engineering

The general problem

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

$$\vec{x}_i = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \ldots \\ x_i^{(d)} \end{bmatrix}$$

- We want to find a good linear prediction rule:

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

The general solution

- Use design matrix

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

  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 $H(x) = w_1 \frac{1}{x^2} + w_2 \sin x + w_3 e^x$?

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**.

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) 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 8482 5148.

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.

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

- 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

- 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

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

  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 $\vec{w}^*$.

Follow along with the demo by clicking the **code** link on the course website next to Lecture 13.

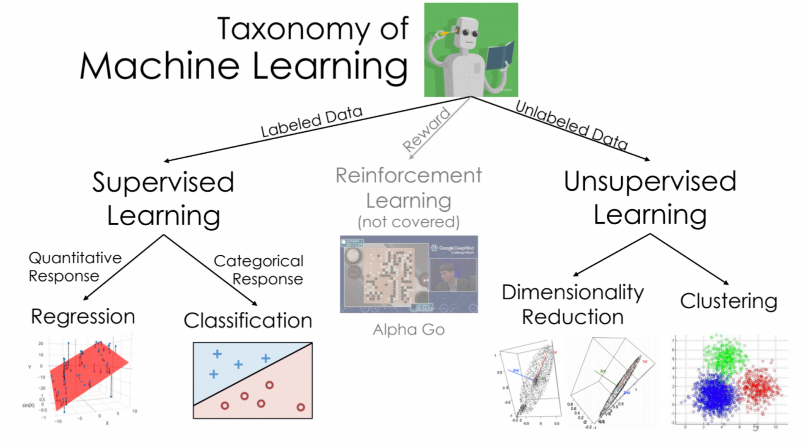
Non-linear prediction rules in general

- 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

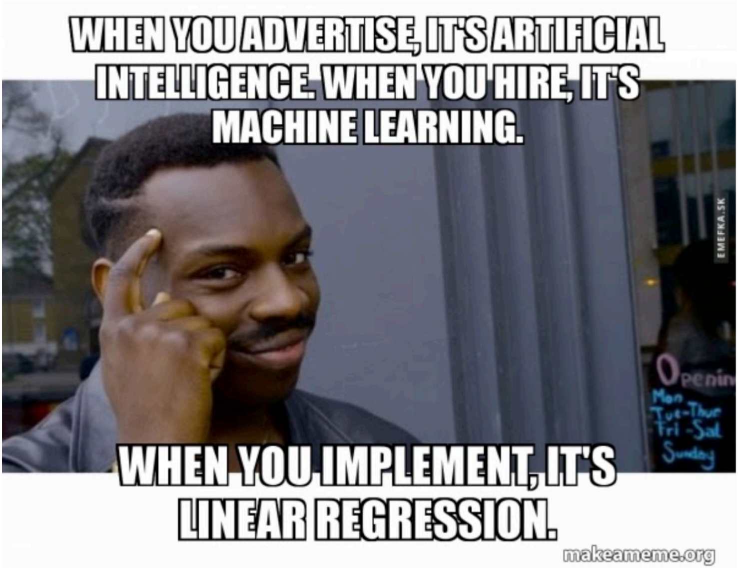
What is machine learning?

- **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 @ UC Berkeley


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

Summary

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.