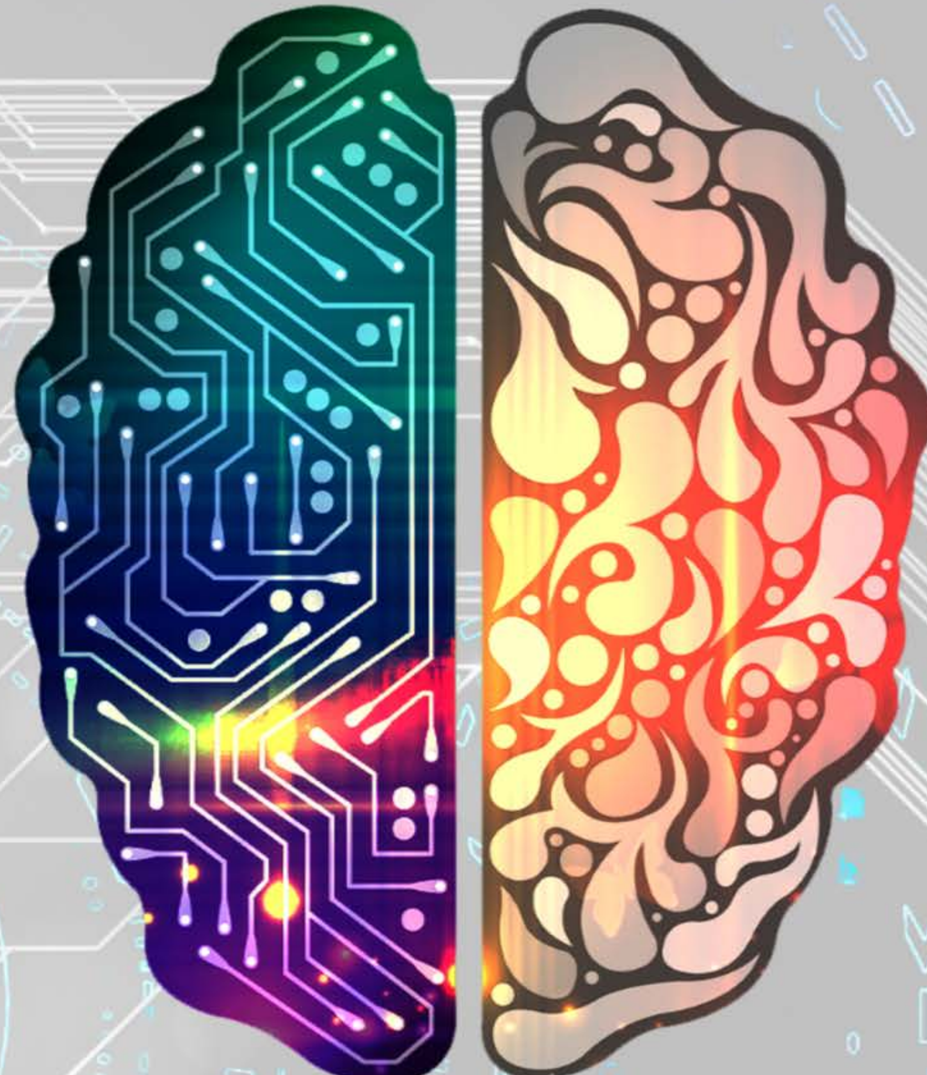
CMPS 460 – Spring 2022

MACHINE LEARNING

Tamer Elsayed



5.a

Practical Issues: Dealing with Features

Sec 2.7,
Sec. 3.1,
Sec 5-5.4

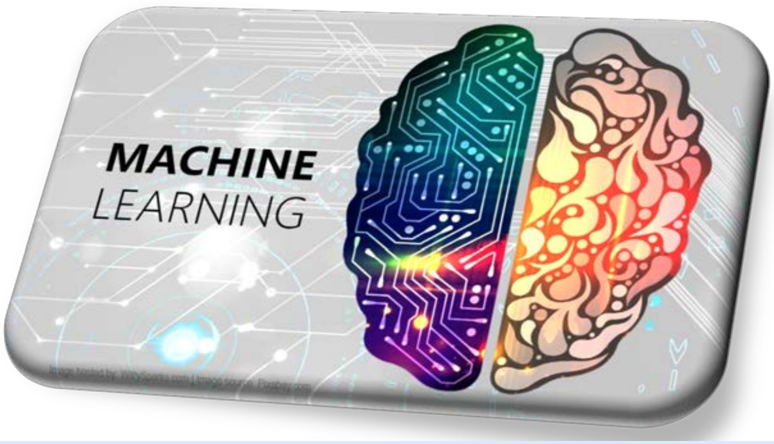
Roadmap …

- Learning algorithm is only one of many steps in designing a ML application
- Practical strategies:
  - Improving features
  - Evaluation
  - Cross Validation
  - Statistical Significance
  - Debugging
- Fundamental ML concepts: estimation vs. approximation error

ML Application …

Typical Design Process for a ML App.

1	real world goal	increase revenue
2	real world mechanism	better ad display
3	learning problem	classify click-through
4	data collection	interaction w/ current system
5	collected data	query, ad, click
6	data representation	bow^2, $\pm$ click
7	select model family	decision trees, depth 20
8	select training data	subset from april'16
9	train model & hyperparams	final decision tree
10	predict on test data	subset from may'16
11	evaluate error	zero/one loss for $\pm$ click
12	deploy!	(hope we achieve our goal)



Dealing with Features

Features, values, vectors …

- To a machine, the **features** themselves have no meaning.
  - Only the **feature values!**

- Feature vector $x = [x_1, x_2, \dots, x_D]$: consisting of one "dimension" for each feature, where each dimension is simply some real value.

[0.1   -3.4   5.0   1.0   …………   0.0   4.1   0.69]

value of feature #3

Feature Types

mapping in a feature vector?

- **Quantitative**
  - Binary: 0/1, -1/1, T/F
  - Real-valued: GPA, temperature, salary
- **Categorical**
  - Sports, health, political, educational, economical
  - Red, green, blue, yellow
- **Ordinal**
  - Low, medium, high
  - Weak, fair, strong, very strong
  - Positive, neutral, negative

How about text?

- Raw text cannot be used directly in ML.

Why?

- Text is composed of a sequence of words.
- Words indicate meaning.

Words as features!

Text Representation: Bag of Words

- Simple representation of text used in NLP and IR.

- Text (e.g., sentence, tweet, article, web page) is represented as the ***bag*** (or a set) ***of*** its ***words***, ignoring its order.

- ***Each word is a feature***
  - the feature value can be binary (appears in the text or not) or frequency of the word in the text (or even better – later).

Example

Given the following collection of documents:

d_1: he likes to wink he likes to drink

d_2: he likes to drink and drink and drink

d_3: the thing he likes to drink is ink

d_4: the ink he likes to drink is pink

d_5: he likes to wink and drink pink ink

Feature vectors: binary values

Vocabulary	d_1	d_2	d_3	d_4	d_5
he	1	1	1	1	1
likes	1	1	1	1	1
to	1	1	1	1	1
wink	1	0	0	0	1
drink	1	1	1	1	1
and	0	1	0	0	1
the	0	0	1	1	0
thing	0	0	1	0	0
ink	0	0	1	1	1
is	0	0	1	1	0
pink	0	0	0	1	1

Feature vectors: frequency values …

Vocabulary	d_1	d_2	d_3	d_4	d_5
he	2	1	1	1	1
likes	2	1	1	1	1
to	2	1	1	1	1
wink	1	0	0	0	1
drink	1	3	1	1	1
and	0	2	0	0	1
the	0	0	1	1	0
thing	0	0	1	0	0
ink	0	0	1	1	1
is	0	0	1	1	0
pink	0	0	0	1	1

Importance of Good Features …

- “Garbage in, Garbage out”!
- Learning algorithms can’t compensate for useless training examples
    - e.g., if all features are irrelevant

- Feature design can have bigger impact on performance than tweaking the learning algorithm
    – e.g., feature combination

Irrelevant & Redundant Features

- Irrelevant features: completely uncorrelated with the prediction task.
  - e.g., the word “the”

- Redundant features: highly correlated, regardless of being relevant or not.
  - e.g., close pixels in images

DT?

kNN?

Perceptron?

Feature Pruning

- Very useful and applied in many applications.

- Easiest in the case of binary features.
  - If appears some small number K times
    - e.g., misspellings
  - If appears in all-but-K times
    - e.g., the word “the”

- For real-valued features
  - look for features with low variance.

Normalization

- Feature normalization

Centering:	$x_{n,d} \leftarrow x_{n,d} - \mu_d$
Variance Scaling:	$x_{n,d} \leftarrow x_{n,d} / \sigma_d$
Absolute Scaling:	$x_{n,d} \leftarrow x_{n,d} / r_d$

where:

$$\mu_d = \frac{1}{N} \sum_n x_{n,d}$$

$$\sigma_d = \sqrt{\frac{1}{N-1} \sum_n (x_{n,d} - \mu_d)^2}$$

$$r_d = \max_n \left| x_{n,d} \right|$$

Min-max Scaling

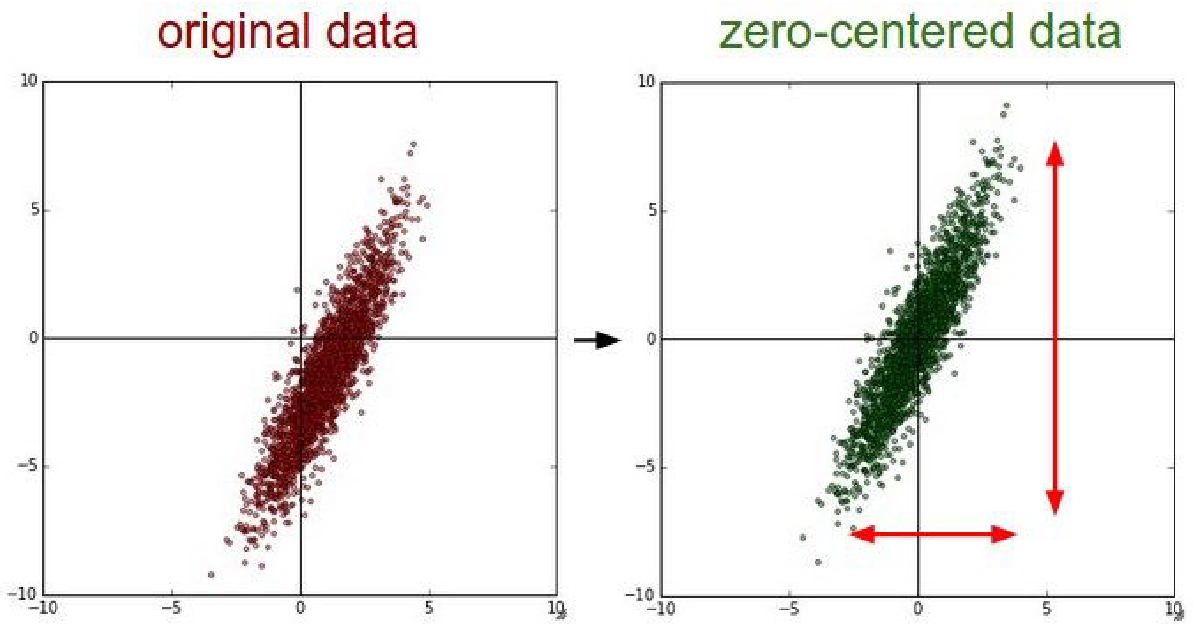
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Example normalization

$$\boldsymbol{x}_n \leftarrow \boldsymbol{x}_n / \, ||\boldsymbol{x}_n||$$

Normalization: Centering

$$x_{n,d} \leftarrow x_{n,d} - \mu_d$$

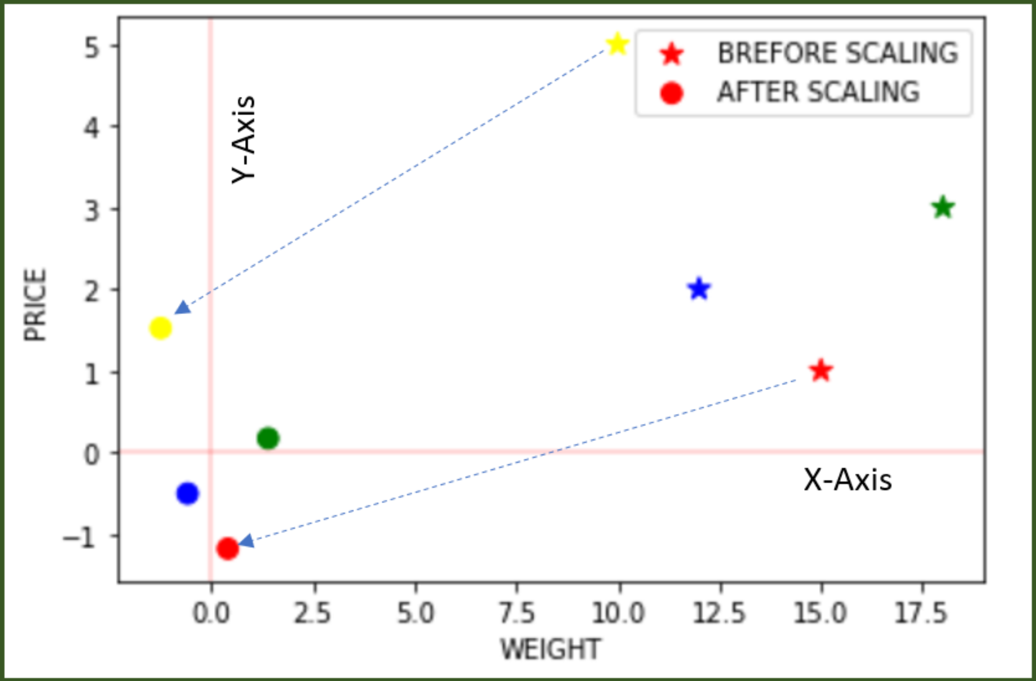


http://luisevalencia.com/feature-normalization/

Normalization: Z-score normalization

Centering: $x_{n,d} \leftarrow x_{n,d} - \mu_d$

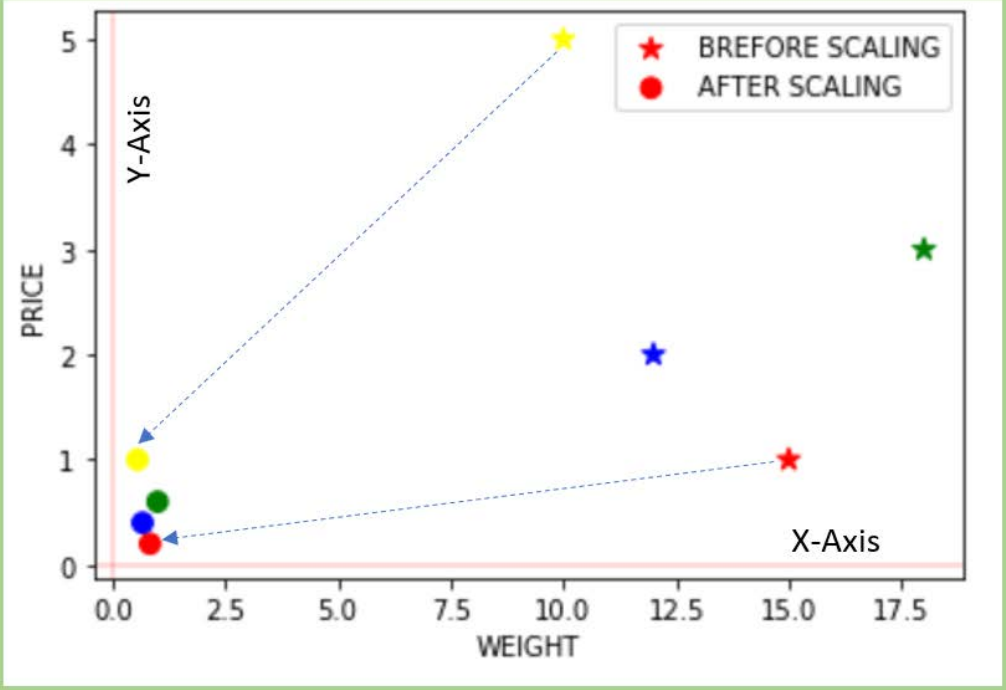
Variance Scaling: $x_{n,d} \leftarrow x_{n,d} / \sigma_d$



https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Normalization: Absolute Scaling

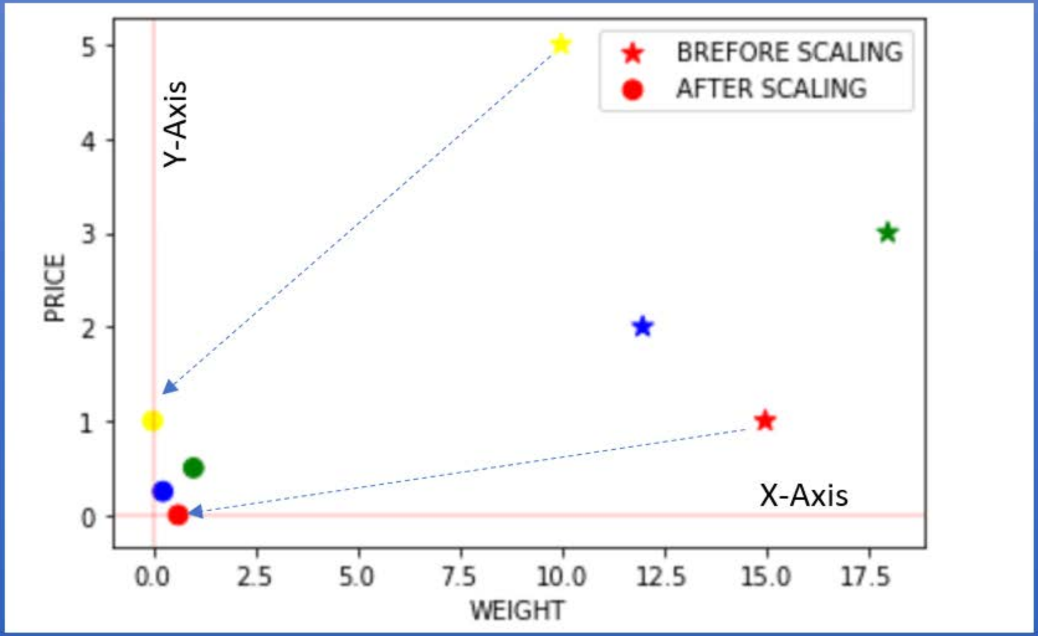
$$x_{n,d} \leftarrow x_{n,d} / r_d$$



https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Normalization: Min-max Scaling

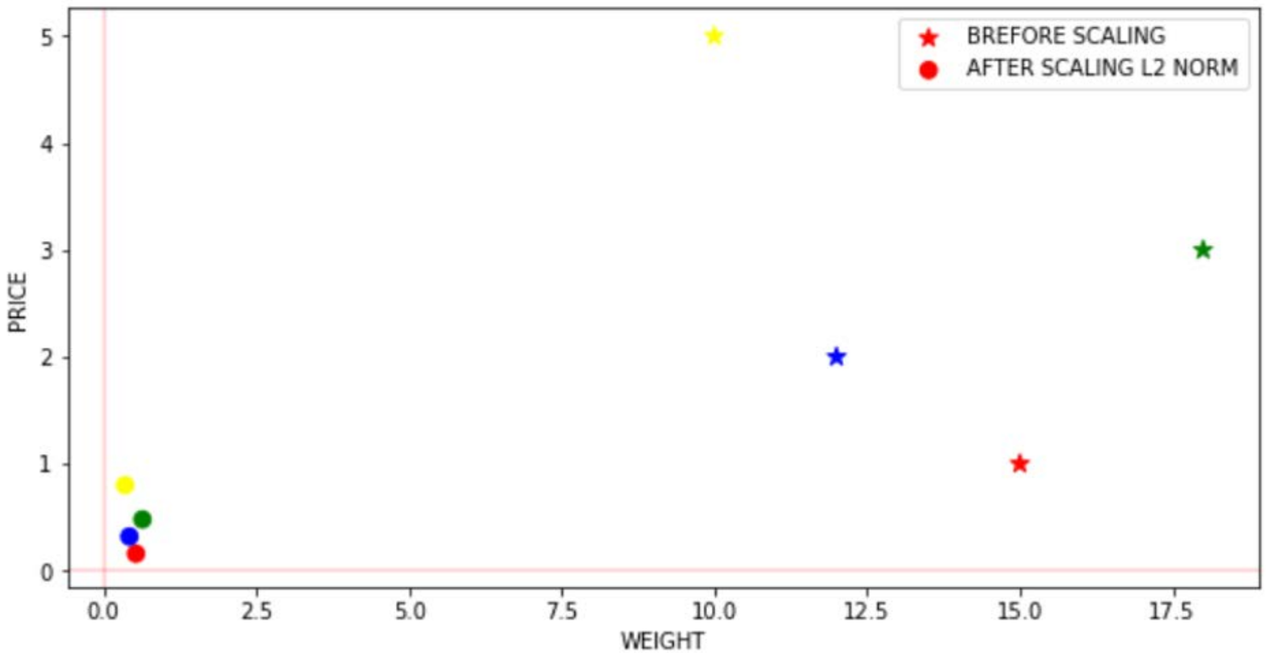
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Normalization: Example Normalization

$$x_n \leftarrow x_n \,/\, ||x_n||$$



https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Logarithmic Transformation

- Mainly for textual features.
- Importance is not linear with frequency

$$x_d \mapsto \log_2(x_d + 1)$$

Feature Combination

- DT is essentially building meta features.

How?

- Constructing meta features for perceptron from DT
    1. train a shallow DT to extract meta features
    2. add only those feature combinations to the feature set for the perceptron.