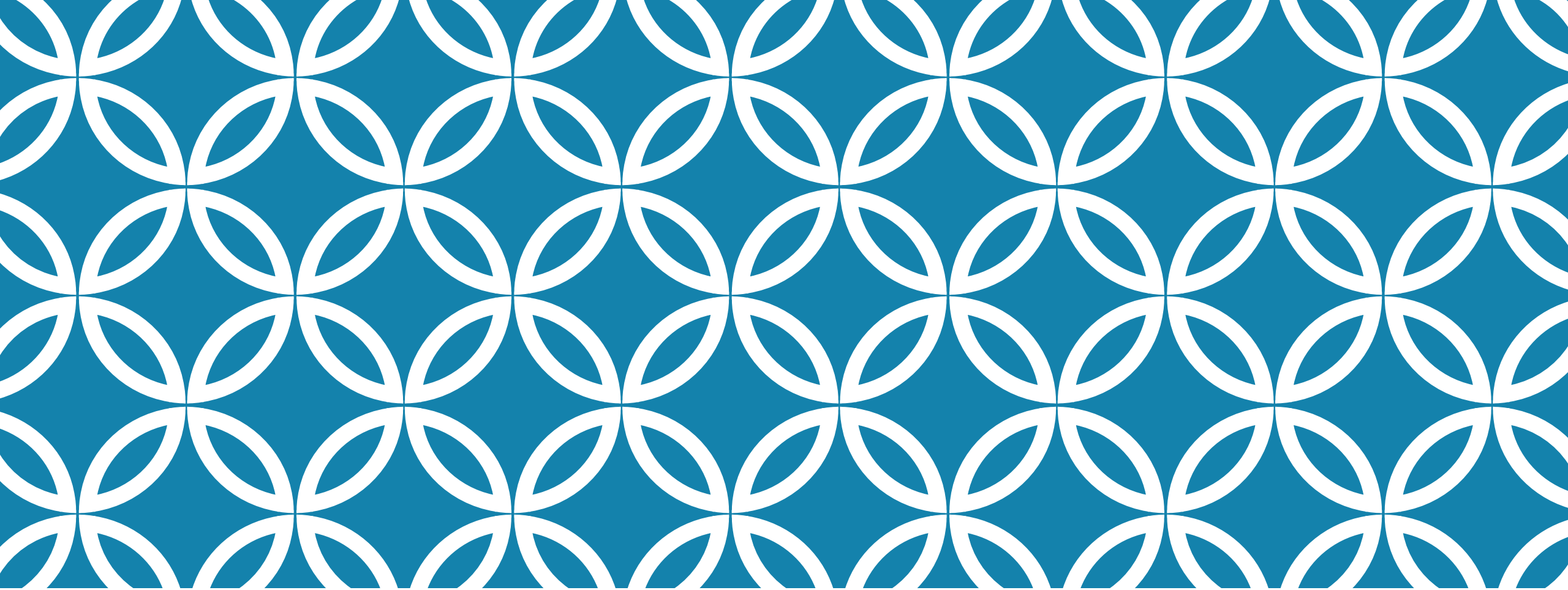


1.4 FEATURE ENGINEERING

CSC 462 [THPMLB:5,HMLSKT:2]

OUTLINE

- “Data Mining” vs. “Machine Learning”
- Sources of Data for Features
- Feature Engineering Overview
- Converting Between Feature Types
- One-Hot Encoding
- Binning
- Normalization
- Scaling
- Dealing with Missing Values

"DATA MINING" VS. "MACHINE LEARNING"

- Machine learning and data mining often employ the same methods and overlap significantly.
- But while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases).
- Data mining uses many machine learning methods, but with different goals.
- On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.
- Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with.
- In machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge.
- Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

SOURCES OF DATA FOR FEATURES

System State

- App in foreground?
- Roaming?
- Sensor readings

Interaction History

- User's 'report as junk' rate
- # previous interactions with sender
- # messages sent/received

Content Analysis

- Stuff we've been talking about
- Stuff we're going to talk about next

User Information

- Industry
- Demographics

Metadata

- Properties of phone #s referenced
- Properties of the sender
- Run other models on the content
  - Grammar
  - Language
  - …

GOALS OF FEATURE ENGINEERING

- Convert ‘context’ -> input to learning algorithm.
- Expose the structure of the concept to the learning algorithm.
- Work well with the structure of the model the algorithm will create.
- Balance number of features, complexity of concept, complexity of model, amount of data.

SAMPLE FROM SMS SPAM

- SMS Message (arbitrary text) -> 5 dimensional array of binary features

- 1 if message is longer than 40 chars, 0 otherwise
- 1 if message contains a digit, 0 otherwise
- 1 if message contains word ‘call’, 0 otherwise
- 1 if message contains word ‘to’, 0 otherwise
- 1 if message contains word ‘your’, 0 otherwise

“SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info”

Long?	HasDigit?	ContainsWord(Call)	ContainsWord(to)	ContainsWord(your)

BASIC FEATURE TYPES

Binary Features

- ContainsWord(call)?
- IsLongSMSMessage?
- Contains(*#)?
- ContainsPunctuation?

Categorical Features

- FirstWordPOS ->
  { Verb, Noun, Other }
- MessageLength ->
  { Short, Medium, Long, VeryLong }
- TokenType ->
  { Number, URL, Word, Phone#, Unknown }
- GrammarAnalysis ->
  - { Fragment, SimpleSentence, ComplexSentence }

Numeric Features

- CountOfWord(call)
- MessageLength
- FirstNumberInMessage
- WritingGradeLevel

CONVERTING BETWEEN FEATURE TYPES

Numeric Feature => Binary Feature — Single threshold

Length of text + [40] => { 0, 1 }

Numeric Feature => Categorical Feature — Set of thresholds

Length of text + [20, 40] => { short or medium or long }

Categorical Feature => Binary Features — One-hot encoding

{ short or medium or long } => [1, 0, 0] or [0, 1, 0] or [0, 0, 1]

Binary Feature => Numeric Feature — …

{ 0, 1 } => { 0, 1 }

ONE-HOT ENCODING

- Some learning algorithms only work with numerical feature vectors.
- Transform categorical feature into several binary ones -> increase the dimensionality of the feature vectors.

> If your example has a categorical feature "colors" and this feature has three possible values: "red," "yellow," "green," you can transform this feature into a vector of three numerical values:

$$\begin{aligned} \text{red} &= [1, 0, 0] \\ \text{yellow} &= [0, 1, 0] \\ \text{green} &= [0, 0, 1] \end{aligned} \tag{1}$$

- You should not transform red into 1, yellow into 2, and green into 3 to avoid increasing the dimensionality because that would imply that there's an order among the values in this category and this specific order is important for the decision making.

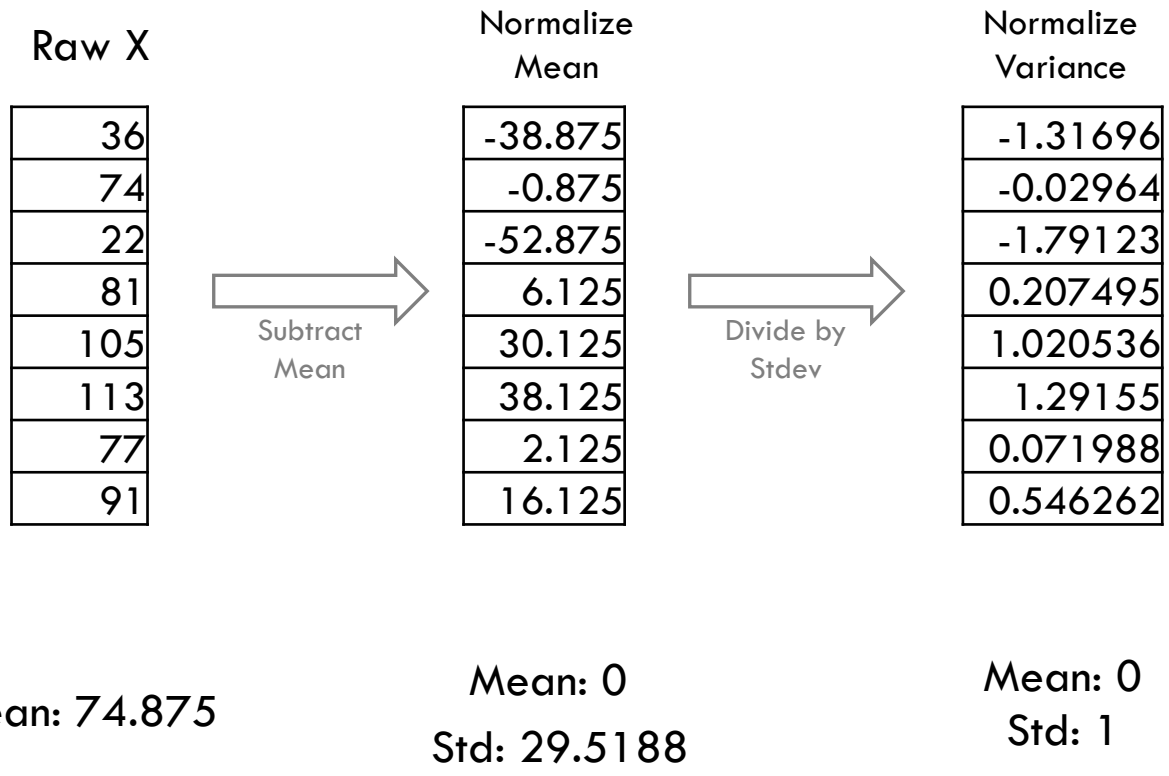
BINNING (BUCKETING)

- When you have a numerical feature but you want to convert it into a categorical one.
- Binning is the process of converting a continuous feature into multiple binary features called bins or buckets, typically based on value range.
- For example, instead of representing age as a single real-valued feature, the analyst could chop ranges of age into discrete bins: all ages between 0 and 5 years-old could be put into one bin, 6 to 10 years-old could be in the second bin, 11 to 15 years-old could be in the third bin, and so on.

```
median_income = tf.feature_column.numeric_column("median_income")
bucketized_income = tf.feature_column.bucketized_column(
    median_income, boundaries=[1.5, 3., 4.5, 6.])
```

If the `median_income` feature is equal to, say, 3.2, then the `bucketized_income` feature will automatically be equal to 2 (i.e., the index of the corresponding income bucket).

NORMALIZATION (NUMERIC => BETTER NUMERIC)



Z-score Normalization

Helps make model's job easier

- No need to learn what is 'big' or 'small' for the feature
- Some model types benefit more than others

To use in practice:

- Estimate mean/stdev on training data
- Apply normalization using those parameters to validation /train

FEATURE NORMALIZATION

- We can **speed up gradient descent** by **having each of our input values in roughly the same range**
- Two techniques to help with this are **feature scaling** and **mean normalization**
  1. **Feature scaling:** involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1.
  2. **Mean normalization:** involves subtracting the average value for an input variable from the values for that input variable, resulting in a new average value for the input variable of just zero.

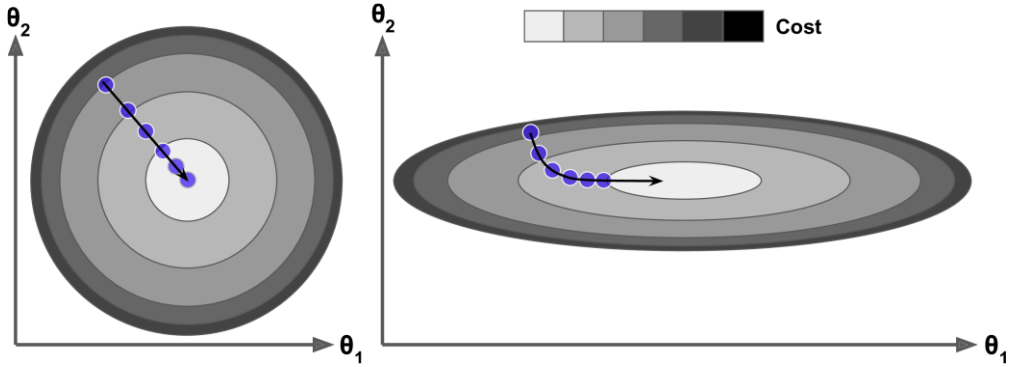
FEATURE SCALING

A significant issue is that the range of the feature may differ a lot. Using the original values may put more weights on the features with a large range.x_1= size (0-2000 feet^2)

x_2= number of bedrooms (1-5)

In order to deal with this problem, we need to apply the technique of feature rescaling (either normalization and standardization) as a pre-processing step (do not apply to $x_0 = 1$).

The goal of applying Feature Scaling is to make sure features are on almost the same scale so that each feature is equally important and make it easier to process by most ML algorithms.



Gradient Descent with and without feature scaling

FEATURE SCALING

- Get every feature into approximately $-1 \leq x_i \leq 1$ range
- Do not apply to $x_0 = 1$
- Good:
  - $0 \leq x_i \leq 3$
  - $-2 \leq x_i \leq 0.5$
  - $-3 \leq x_i \leq 3$
  - $-0.5 \leq x_i \leq 0.3$
- Bad:
  - $-100 \leq x_i \leq 100$
  - $-0.00001 \leq x_i \leq 0.00001$

1. MEAN NORMALIZATION

- Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean

$$x_i = \frac{x_i - \mu_i}{\max_i - \min_i}$$

- Do not apply to $x_0 = 1$
- μ_i is the average value of x_i in the training set
- You then divide by the range (i.e. the maximum value minus the minimum value) of the feature
- $x_1 = \frac{size\ (feet)^2 - \mu_1}{2000 - 0}$, e.g. $x_1 = \frac{size\ (feet)^2 - 1000}{2000}$
- $x_2 =$

2. MIN-MAX NORMALIZATION

- Normalization (AKA min-max scaling) is the process of converting an actual range of values which a numerical feature can take, into a standard range of values, typically in the interval [−1, 1] or [0, 1].

$$x_i = \frac{x_i - \min_i}{\max_i - \min_i}$$

where $\min(j)$ and $\max(j)$ are, respectively, the minimum and the maximum value of the feature j in the dataset.

- Why do we normalize? Normalizing the data is not a strict requirement. However, in practice, it can lead to an increased speed of the training process.
- Additionally, it's useful to ensure that our inputs are roughly in the same relatively small range to avoid problems related to working with very small or very big numbers (known as numerical overflow).

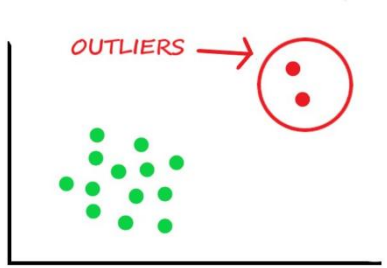
3. STANDARDIZATION

- **Standardization** (or **z-score normalization**) is the procedure during which the feature values are rescaled so that they have the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 1$, where μ is the mean (the average value of the feature, averaged over all examples in the dataset) and σ is the standard deviation from the mean.
- Standard scores (or z-scores) of features are calculated as follows:

$$x_i = \frac{x_i - \mu_i}{\sigma_i}$$

NORMALIZATION VS STANDARDIZATION

- When you should use normalization and when standardization. Usually, if your dataset is not too big and you have time, you can try both and see which one performs better for your task.
- If you don't have time:
  - unsupervised learning algorithms, in practice, more often benefit from standardization than from normalization;
  - standardization is also preferred for a feature if the values this feature takes are distributed close to a normal distribution (so-called bell curve);
  - standardization is preferred for a feature if it can sometimes have extremely high or low values (**outliers**);
  - In all other cases, normalization is preferable.



DEALING WITH MISSING FEATURES

- In some cases, values of some features can be missing.
- That often happens when the dataset was handcrafted, and the person working on it forgot to fill some values or didn't get them measured at all.
- The typical approaches of dealing with missing values for a feature include:
  1. Removing the whole attribute if most of its values are missing.
  2. Removing the examples with missing features from the dataset (that can be done if your dataset is big enough so you can sacrifice some training examples);
  3. Using a learning algorithm that can deal with missing feature values (depends on the library and a specific implementation of the algorithm);
  4. Using a data imputation technique.

Reference: Garcá-Laencina, P. J., Sancho-Gómez, J.-L. & Figueiras-Vidal, A. R. (2010), 'Pattern classification with missing data: a review', Neural Computing and Applications 19(2), 263–282.

DATA IMPUTATION

Data imputation means substituting the missing value. There are several techniques including the following.

1. Statistical imputation methods.
2. Imputation based on machine learning methods.
3. Model-based methods.

STATISTICAL DATA IMPUTATION METHODS

- **Mean imputation:** For each attribute having missing values, an alternative value can be formed using methods such as the following.
  - **Global average–global mode (GA–GM)** method. If the attribute **is numeric, then the average value is computed,** and used to replace each missing value of that attribute. **Otherwise, the mode (the most common value)** is used in the case of nominal attributes.
  - **Concept average–concept mode (CA–CM)** method is a more sophisticated version of the GA–GM method. It works in a similar fashion, however, **the average (and mode) is computed per concept**. That is, to use the same average (or mode) **for instances that belong to the same class** of the instance with a missing value at hand.
- **Regression imputation:** Regression methods are suited when there is a **correlation** between available and missing variables. A **regression model is trained to approximate the missing value,** where the outcome is the missing variable, and the input is all the remaining available variables.
- **Hot and cold deck imputation:**
  - In **hot deck** imputation, **the case that is most similar** (according to some similarity measure) to the one with missing value(s) is identified. Missing values are substituted with the same corresponding values in the most similar case.
  - In **cold deck** method, the search for a most similar case is performed using a different data source.
  - The problem with this method is that **the imputation is based on a single case ignoring global properties of the dataset.**
- **Multiple imputation:** This method reflects the uncertainty associated with a missing value. **Multiple datasets are generated, each with a possible value of the missing one.** Each of the datasets are analyzed and then results are combined in some manner (for example see (Grzymala-Busse 1991)). For numeric attributes, the possible values can be the associated discretization intervals.

IMPUTATION BASED ON MACHINE LEARNING METHODS

- **A model is built to predict the missing value** in a certain feature based on available features.
- Different machine learning methods are used including k-NN, SVM, and ANN.
- **A separate model** has to be built **for each attribute having missing values.**
- These methods are generally **robust,** however the main drawback is their **high computational cost.**

MODEL-BASED METHODS

- These methods (Dempster et al. 1977, Gourraud et al. 2004, Schneider 2001) are based on assumptions about the statistical probability distribution of the variables in the model.
- In the expectation-maximization (EM) algorithm, the parameters of the probability distribution model for the incomplete dataset are estimated.
- An iterative process of two steps follows.
  - In the E-step, missing values are imputed based on their conditional expectation values.
  - In the M-step, the parameters are re-estimated based on the complete dataset.
- The two steps are iterated until the imputed values and the estimated parameters stop changing markedly from one cycle to another.