Architecture of ML Systems
09 Data Acquisition and Preparation

Matthias Boehm
Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

Last update: May 29, 2020
Announcements/Org

#1 Video Recording
- Link in TeachCenter & TUbe (lectures will be public)
- Live streaming through TUbe, starting May 08
- Questions: https://tugraz.webex.com/meet/m.boehm

#2 AMLS Programming Projects
- Status: all project discussions w/ 15 students (~8 PRs)
- Awesome mix of projects (algorithms, compiler, runtime)
- Soft deadline: June 30

#3 TU Delft DESOSO 2020
- Delft Students on Software Architecture (incl ML systems) https://desosa.nl
Recap: The Data Science Lifecycle

Data Science Lifecycle

Data-centric View:
- Application perspective
- Workload perspective
- System perspective

Data/SW Engineer

Data Integration
- Data Cleaning
- Data Preparation

Model Selection
- Training
- Hyper-parameters

Validate & Debug
- Deployment
- Scoring & Feedback

Exploratory Process
(experimentation, refinements, ML pipelines)

DevOps Engineer

Data Scientist
The 80% Argument

- **Data Sourcing Effort**
  - Data scientists spend *80-90% time* on finding, integrating, cleaning datasets

- **Technical Debts in ML Systems**
  - Glue code, pipeline jungles, dead code paths
  - Plain-old-data types (arrays), multiple languages, prototypes
  - Abstraction and configuration debts
  - Data testing, reproducibility, process management, and cultural debts

---


Agenda

- Data Acquisition and Integration
- Data Preparation and Feature Engineering
- Data Transformation and Cleaning
- Data Augmentation (next week)

“least enjoyable tasks in data science lifecycle”
Data Acquisition and Integration

Data Integration for ML and ML for Data Integration

Data Integration and Large-Scale Analysis (DIA)
(bachelor/master)
Data Sources and Heterogeneity

- **Terminology**
  - **Integration** (Latin integer = whole): consolidation of data objects / sources
  - **Homogeneity** (Greek homo/homoios = same): similarity
  - **Heterogeneity**: dissimilarity, different representation / meaning

- **Heterogeneous IT Infrastructure**
  - Common enterprise IT infrastructure contains >100s of heterogeneous and distributed systems and applications
  - E.g., health care data management: 20 - 120 systems

- **Multi-Modal Data (example health care)**
  - **Structured patient data**, patient records incl. prescribed drugs
  - **Knowledge base** drug APIs (active pharmaceutical ingredients) + interactions
  - **Doctor notes** (text), diagnostic codes, outcomes
  - **Radiology images** (e.g., MRI scans), **patient videos**
  - **Time series** (e.g., EEG, ECoG, heart rate, blood pressure)
Types of Data Formats

- **General-Purpose Formats**
  - CSV (comma separated values), JSON (javascript object notation), XML, Protobuf
  - CLI/API access to DBs, KV-stores, doc-stores, time series DBs, etc

- **Sparse Matrix Formats**
  - Matrix market: text IJV (row, col, value)
  - Libsvm: text compressed sparse rows
  - Scientific formats: NetCDF, HDF5

- **Large-Scale Data Formats**
  - Parquet (columnar file format)
  - Arrow (cross-platform columnar in-memory data)

- **Domain-Specific Formats**
  - Health care: DICOM images, HL7 messages (health-level seven, XML)
  - Automotive: MDF (measurements), CDF (calibrations), ADF (auto-lead XML)
  - Smart production: OPC (open platform communications)
Types of Heterogeneity

Heterogeneity

Semantic Heterogeneity

1. Synonyms/homonyms
2. Simple mapping (mathematical)
3. Different data types
4. Complex mappings
5. Language expressions

Attribute Heterogeneity

6. Nulls (Missing Values)
7. Virtual columns
8. Semantic incompatibility

Missing Data

Structural Heterogeneity

9. Same attribute in different structure
10. Handling Sets
11. Attribute name w/o semantics
12. Attribute composition

Identification of Data Sources

- **Data Catalogs**
  - Data curation in repositories for finding relevant datasets in **data lakes**
  - Augment data with open and linked data sources

- **Examples**

  **SAP Data Hub**

  ![SAP Data Hub slide]

  ![SAP Sapphire Now 2019]

  **Google Data Search**

  ![Google Data Search slide]

Schema Detection and Integration

- **Schema Detection**
  - Sample of the input dataset → infer *syntactic schema* (e.g., data types)
  - *Semantic schema* detection (e.g., location, date, rank, name)

- **Schema Matching**
  - Semi-automatic mapping of schema S1 to schema S2
  - *Approaches*: Schema- vs instance-based; element- vs structure-based; linguistic vs rules
  - Hybrid and composite matchers
  - Global schema matching (one-to-one): stable marriage problem

- **Schema Mapping**
  - Given two schemas and correspondences, generate transformation program
  - *Challenges*: complex mappings (1:N cardinality), new values, PK-FK relations and nesting, creation of duplicates, different data types, semantic preserving

[Credit: Erhard Rahm]
Corrupted Data

- **Heterogeneity of Data Sources**
  - Update anomalies on denormalized data / eventual consistency
  - Changes of app/preprocessing over time (US vs us) → inconsistencies

- **Human Error**
  - Errors in semi-manual data collection, laziness (see default values), bias
  - Errors in data labeling (especially if large-scale: crowd workers / users)

- **Measurement/Processing Errors**
  - Unreliable HW/SW and measurement equipment (e.g., batteries)
  - Harsh environments (temperature, movement) → aging

---

### Table: Data Acquisition and Integration

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>BDay</th>
<th>Age</th>
<th>Sex</th>
<th>Phone</th>
<th>Zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Smith, Jane</td>
<td>05/06/1975</td>
<td>44</td>
<td>F</td>
<td>999-9999</td>
<td>98120</td>
</tr>
<tr>
<td>3</td>
<td>John Smith</td>
<td>38/12/1963</td>
<td>55</td>
<td>M</td>
<td>867-4511</td>
<td>11111</td>
</tr>
<tr>
<td>7</td>
<td>Jane Smith</td>
<td>05/06/1975</td>
<td>24</td>
<td>F</td>
<td>567-3211</td>
<td>98120</td>
</tr>
</tbody>
</table>

### Notes:
- **Uniqueness & duplicates**: Contradictions & wrong values
- **Missing Values**: Ref. Integrity

[Credit: Felix Naumann]

- **Typos**:
  - 98120: San Jose
  - 90001: Lost Angeles
Examples (aka errors are everywhere)

- **DM SS’19**
  *(Soccer World Cups)*

- **DM WS’19/20**
  *(Airports and Airlines)*

- **DM SS’20**
  *(DBLP Publications)*
Data Integration for ML and ML for DI

- **#1 Data Extraction**
  - Extracting structured data from un/semi-structured data
  - Rule- and ML-based extractors, combination w/ CNN

- **#2 Schema Alignment**
  - Schema matching for consolidating data from heterogeneous systems
  - Spatial and Temporal alignment via provenance and query processing (e.g., sensor readings for object along a production pipeline)

- **#3 Entity Linking**
  - Linking records to entities (deduplication)
  - Blocking, pairwise matching, clustering, ML, Deep ML (via entity embedding)

- **#4 Data Fusion**
  - Resolve conflicts, necessary in presence of erroneous data
  - Rule- and ML-based, probabilistic GM, Deep ML (RBMs, graph embeddings)

[Xin Luna Dong, Theodoros Rekatsinas: Data Integration and Machine Learning: A Natural Synergy. SIGMOD 2018]
Data Validation

Sanity checks on expected shape before training first model

- Check a feature’s min, max, and most common value
  - Ex: Latitude values must be within the range [-90, 90] or [-\(\pi/2, \pi/2\)]

- The histograms of continuous or categorical values are as expected
  - Ex: There are similar numbers of positive and negative labels

- Whether a feature is present in enough examples
  - Ex: Country code must be in at least 70% of the examples

- Whether a feature has the right number of values (i.e., cardinality)
  - Ex: There cannot be more than one age of a person

Data Validation, cont.

- **Constraints and Metrics for quality check UDFs**

<table>
<thead>
<tr>
<th>constraint</th>
<th>arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimension completeness</td>
<td>column</td>
</tr>
<tr>
<td>isComplete</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasCompleteness</td>
<td>column, udf</td>
</tr>
<tr>
<td>dimension consistency</td>
<td>column, column, value range</td>
</tr>
<tr>
<td>isUnique</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasUniqueness</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasDistinctness</td>
<td>column, udf</td>
</tr>
<tr>
<td>isInRange</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasConsistentType</td>
<td>column, udf</td>
</tr>
<tr>
<td>isNonNegative</td>
<td>column, column pair</td>
</tr>
<tr>
<td>satisfies</td>
<td>predicate</td>
</tr>
<tr>
<td>satisfiesIf</td>
<td>predicate pair</td>
</tr>
<tr>
<td>hasPredictability</td>
<td>column, column(s), udf</td>
</tr>
<tr>
<td>statistics (can be used to verify dimension consistency)</td>
<td>udf</td>
</tr>
<tr>
<td>hasSize</td>
<td>udf</td>
</tr>
<tr>
<td>hasTypeConsistency</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasCountDistinct</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasApproxCountDistinct</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasMin</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasMax</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasMean</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasStandardDeviation</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasApproxQuantile</td>
<td>column, quantile, udf</td>
</tr>
<tr>
<td>hasEntropy</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasMutualInformation</td>
<td>column, column pair</td>
</tr>
<tr>
<td>hasHistogramValues</td>
<td>column, udf</td>
</tr>
<tr>
<td>hasCorrelation</td>
<td>column, column pair</td>
</tr>
<tr>
<td>time</td>
<td></td>
</tr>
<tr>
<td>hasNoAnomalies</td>
<td>metric, detector</td>
</tr>
<tr>
<td>metric</td>
<td>dimension completeness</td>
</tr>
<tr>
<td></td>
<td>Completeness</td>
</tr>
<tr>
<td></td>
<td>dimension consistency</td>
</tr>
<tr>
<td></td>
<td>Size</td>
</tr>
<tr>
<td></td>
<td>Compliance</td>
</tr>
<tr>
<td></td>
<td>Uniqueness</td>
</tr>
<tr>
<td></td>
<td>Distinctness</td>
</tr>
<tr>
<td></td>
<td>ValueRange</td>
</tr>
<tr>
<td></td>
<td>DataTypes</td>
</tr>
<tr>
<td></td>
<td>Predictability</td>
</tr>
</tbody>
</table>

- **Approach**
  - #1 Quality checks on basic metrics, computed in **Apache Spark**
  - #2 **Incremental maintenance** of metrics and quality checks

[Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Bießmann, Andreas Grafberger: Automating Large-Scale Data Quality Verification. *PVLDB 2018*]

Organizational Lesson: benefit of shared vocabulary/procedures

Technical Lesson: fast/scalable; reduce manual and ad-hoc analysis
Data Preparation and Feature Engineering
Overview Feature Engineering

- **Terminology**
  - Matrix $X$ of $m$ observations (rows) and $n$ features (columns)
  - **Continuous features**: numerical values (aka scale features)
  - **Categorical features**: non-numerical values, represent groups
  - **Ordinal features**: non-numerical values, associated ranking
  - Feature space: multi-dimensional space of features $\rightarrow$ curse of dimensionality

- **Feature Engineering**
  - Bring multi-modal data and features into numeric representation
  - Use domain expertise to expose predictive features to ML model training

- **Excursus: Representation Learning**
  - Neural networks can be viewed as combined representation learning and model training (pros and cons: learned, repeatable)
  - Mostly homogeneous inputs (e.g., image), research on multi-modal learning

** Principle:** If same accuracy, prefer simple model (cheap, robust, explainable)
Recoding

Summary

- Numerical encoding of categorical features (arbitrary strings)
- Map distinct values to integer domain (potentially combined w/ one-hot)

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
<th>Dictionaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Jose</td>
<td>CA</td>
<td>{San Jose : 1, New York : 2, San Francisco : 3, Seattle : 4, Boston : 5, Los Angeles : 6}</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
<td></td>
</tr>
<tr>
<td>San Francisco</td>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>WA</td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
<td></td>
</tr>
<tr>
<td>Boston</td>
<td>MA</td>
<td></td>
</tr>
<tr>
<td>San Francisco</td>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>Los Angeles</td>
<td>CA</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>WA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Feature Hashing

- **Summary**
  - Numerical encoding of categorical features (arbitrary strings)
  - Hash input to k buckets via hash(value) % k (often combined w/ one-hot)

<table>
<thead>
<tr>
<th>City</th>
<th>For k = 5:</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Jose</td>
<td>1993955031 % 5 → 1</td>
<td>1</td>
</tr>
<tr>
<td>New York</td>
<td>1382994575 % 5 → 0</td>
<td>0</td>
</tr>
<tr>
<td>San Francisco</td>
<td>1540367136 % 5 → 1</td>
<td>1</td>
</tr>
<tr>
<td>Seattle</td>
<td>-661909336 % 5 → 1</td>
<td>1</td>
</tr>
<tr>
<td>New York</td>
<td>1993955031 % 5 → 1</td>
<td>1</td>
</tr>
<tr>
<td>Boston</td>
<td>1995575789 % 5 → 4</td>
<td>4</td>
</tr>
<tr>
<td>San Francisco</td>
<td>1540367136 % 5 → 1</td>
<td>1</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>-425347233 % 5 → 3</td>
<td>3</td>
</tr>
<tr>
<td>Seattle</td>
<td>-661909336 % 5 → 1</td>
<td>1</td>
</tr>
</tbody>
</table>

Efficient, but collisions
Binning (see also Quantization, Binarization)

- **Summary**
  - Encode of numerical features to integer domain (often combined w/ one-hot)
  - **Equi-width**: split (max-min)-range into k equal-sized buckets
  - **Equi-height**: compute data-driven ranges for k balanced buckets

<table>
<thead>
<tr>
<th>Sqft</th>
<th>Sqft-Bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>928.5</td>
<td>2</td>
</tr>
<tr>
<td>451</td>
<td>1</td>
</tr>
<tr>
<td>570.3</td>
<td>1</td>
</tr>
<tr>
<td>1,273</td>
<td>3</td>
</tr>
<tr>
<td>1,239</td>
<td>3</td>
</tr>
<tr>
<td>711.3</td>
<td>1</td>
</tr>
<tr>
<td>1,114</td>
<td>3</td>
</tr>
<tr>
<td>867</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Equal-sized numerical buckets (with k=3)**
  - min = 451
  - max = 1,273
  - range = 822
  - [451, 725) → 1
  - [725, 999) → 2
  - [999, 1,273] → 3

- Allows modelling small, medium, large apartments
## One-hot Encoding

### Summary
- Encode integer feature of cardinality $d$ into sparse 0/1 vector of length $d$
- Feature vectors of input features concatenated in sequence

<table>
<thead>
<tr>
<th>City</th>
<th>State</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Derived Features

- **Intercept Computation**
  - Add a column of ones to X for computing the intercept as a weight
  - Applies to regression and classification

- **Non-Linear Relationships**
  - Can be explicitly materialized as feature combinations
  - Example: Assumptions of underlying physical system
  - Arbitrary complex feature interactions: e.g., $X_1^2 \times X_2$

```
X = cbind(X, matrix(1, nrow(X), 1));  // y ~ b1*X1 + b2*X1^2
X = cbind(X, X^2);                     // y ~ b1*X1 + b2*X1^2
```
NLP Features

- **Basic NLP Feature Extraction**
  - *Sentence/word tokenization:* split into sentences/words (e.g., via stop words)
  - *Part of Speech (PoS) tagging:* label words verb, noun, adjectives (syntactic)
  - *Semantic role labeling:* label entities with their roles in actions (semantic)
    
    Who did *what* to *whom* at *where*?

- **Bag of Words (BOW) and N-Grams**
  - Represent sentences as *bag* (multisets)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Bi-grams:** bag-of-words for 2-sequences of words (order preserving)
- **N-grams:** generalization of bi-grams to arbitrary-length sequences
NLP Features, cont.

- **Word Embeddings**
  - Trained (word → vector) mappings (≈ 50-300 dims)
  - **Word2vec**: continuous bag-of-words (CBOW) or continuous skip-gram
  - Subsampling frequent words
  - **Semantic preserving arithmetic operations** (+ ~ * of context distributions)

- **Follow-up Work**
  - Often pre-trained word embeddings; fine-tuning if necessary for task/domain
  - Various extensions/advancements: **Sentence2Vec**, **Doc2Vec**, **Node2Vec**
  - **BERT, RoBERTa, ALBERT, StructBERT**

[Note: For more information on Word2Vec, see [Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space. ICLR (Workshop) 2013](http://google.com)]

[Note: For more information on BERT, RoBERTa, ALBERT, StructBERT, see [Jacob Devlin et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT (1) 2019](http://google.com)]
Example Spark ML

- **API Design**
  - **Transformers:** Feature transformations and learned models
  - **Estimators:** Algorithm that can be fit to produce a transformer
  - Compose ML pipelines from chains of transformers and estimators

- **Example Pipeline**

```scala
// define pipeline stages
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.outputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.001)

// create pipeline transformer via fit
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
model = pipeline.fit(training)

// use of resulting ML pipeline
prediction = model.transform(test)
```

[https://spark.apache.org/docs/2.4.3/ml-pipeline.html](https://spark.apache.org/docs/2.4.3/ml-pipeline.html)
Example SystemML/SystemDS

- Feature Transformation during Training

```r
# read tokenized words
FX = read("./input/FX", data_type=FRAME); # sentence id, word, count
FY = read("./input/FY", data_type=FRAME); # sentence id, labels

# encode and one-hot encoding
[X0, MX] = transformencode(target=FX, spec="{recode:[2]}”);
[Y0, MY] = transformencode(target=FY, spec="{recode:[2]}”);
X = table(X0[,1], X0[,2], X0[,3]); # bag of words
Y = table(Y0[,1], Y0[,2]); # bag of words

# model training via multi-label, multi-nominal logical regression
B = mlogreg(X, Y);
```
Example SystemML/SystemDS, cont.

- Feature Transformation during Scoring

```
# read tokenized words of test sentences
dFX = read("./input/dFX", data_type=FRAME);  # sentence id, word, count

# encode and one-hot encoding
dX0 = transformapply(target=dFX, spec="{recode:[2]}", meta=MX);
dX = table(dX0[,1], dX0[,2], dX0[,3]);  # bag of words

# model scoring and postprocessing (reshape, attach sentence ID, etc)
dYhat = (X %% B) >= theta; ...;

# decode output labels: sentence id, label word
dFYhat = transformdecode(target=dYhat, spec="{recode:[2]}", meta=MY);
```
Data Transformation and Cleaning
Standardization/Normalization

#1 Standardization
- Centering and scaling to mean 0 and variance 1
- Ensures well-behaved training
- Densifying operation
- Awareness of NaNs
- Batch normalization in DNN: standardization of activations

\[
\begin{align*}
X &= X - \text{colMeans}(X); \\
X &= X / \text{sqrt}(\text{colVars}(X)); \\
X &= \text{replace}(X, \text{pattern}=\text{NaN}, \text{replacement}=0); \ #\text{robustness}
\end{align*}
\]

#2 (Min-Max) Normalization
- Rescale values into common range [0,1]
- Avoid bias to large-scale features
- Does not handle outliers

\[
X = (X - \text{colMins}(X)) / (\text{colMaxs}(X) - \text{colMins}(X));
\]

Recommended Reading
Standardization/Normalization, cont.

- **#3 Deferred Standardization**
  - Avoid densifying dataset upfront by pushing standardization into inner loop iterations
  - Let *matrix-multiplication chain optimization* + rewrites do the rest

- **Example**
  - GLM/lmCG

  Input w/ column of ones (intercept)

  ![Diagram](image)

  **# operation w/ early standardized X**
  
  \[
  q = t(X) \cdot \text{diag}(w) \cdot X \cdot B;
  \]

  **Substitute X with X \cdot S**

  **# operation w/ deferred standardization**
  
  \[
  q = t(S) \cdot t(X) \cdot \text{diag}(w) \cdot X \cdot S \cdot B;
  \]

  \[
  q = t(S) \cdot t(X) \cdot (\text{diag}(w) \cdot (X \cdot S \cdot B));
  \]
Winsorizing and Trimming

- Recap: Quantiles
  - Quantile $Q_p$ w/ $p \in (0,1)$ defined as $P[X \leq x] = p$

- Winsorizing
  - Replace tails of data distribution at user-specified threshold
  - Quantiles / std-dev
    - Reduce skew

- Truncation/Trimming
  - Remove tails of data distribution at user-specified threshold

- Largest Difference from Mean

```r
# compute quantiles for lower and upper
ql = quantile(X, 0.05);
qu = quantile(X, 0.95);

# replace values outside [ql,qu] w/ ql and qu
Y = ifelse(X < ql, ql, X);
Y = ifelse(Y > qu, qu, Y);

# remove values outside [ql,qu]
I = X < qu | X > ql;
Y = removeEmpty(X, "rows", select = I);

# determine largest diff from mean
I = (colMaxs(X) - colMeans(X))
  > (colMeans(X) - colMins(X));
Y = ifelse(xor(I,op), colMaxs(X), colMins(X));
```

[Credit: https://en.wikipedia.org]
Outliers and Outlier Detection

- Types of Outliers
  - **Point outliers**: single data points far from the data distribution
  - **Contextual outliers**: noise or other systematic anomalies in data
  - **Sequence (contextual) outliers**: sequence of values with abnormal shape/agg
  - Univariate vs multivariate analysis
  - Beware of underlying assumptions (distributions)

- Types of Outlier Detection
  - **Type 1 Unsupervised**: No prior knowledge of data, similar to unsupervised clustering
    \[ \rightarrow \text{expectations: distance, # errors} \]
  - **Type 2 Supervised**: Labeled normal and abnormal data, similar to supervised classification
  - **Type 3 Normal Model**: Represent normal behavior, similar to pattern recognition
    \[ \rightarrow \text{expectations: rules/constraints} \]


Missing Value Imputation

- **Missing Value**
  - Application context defines if 0 is missing value or not
  - If differences between 0 and missing values, use NA or NaN

- **Basic Value Imputation**
  - General-purpose: replace by user-specified constant
  - **Continuous variables**: replace by mean
  - **Categorical variables**: replace by median or mode

- **Iterative Algorithms** *(chained-equation imputation)*
  - Train ML model to predict missing information (feature k → label, split data into observed/missing)
  - Noise reduction: feature subsets + averaging

- **Dynamic Imputation**
  - Data exploration w/ on-the-fly imputation
  - Optimal placement of imputation operations

[Jose Cambronero, John K. Feser, Micah Smith, Samuel Madden: Query Optimization for Dynamic Imputation. PVLDB 2017]
Excursus: Time Series Recovery

- **Motivating Use Case**
  - Given overlapping weekly aggregates $y$ (daily moving average)
  - Reconstruct the original time series $X$

- **Problem Formulation**
  - Aggregates $y$
  - Original time series $X$ (unknown)
  - Mapping $O$ of subsets of $X$ to $y$
  - $\Rightarrow$ Least squares regression problem

- **Advanced Method**
  - Discrete Cosine Transform (DCT) (sparsest spectral representation)
  - Non-negativity and smoothness constraints

- **Use case:** high-precision sensor fusion w/ different data granularity

Selected Research Prototypes

- **ActiveClean (SampleClean)**
  - Suggest sample of data for manual cleaning (rule/ML-based detectors, *Simpson's paradox*)
  - Update dirty model with gradients of cleaned data (weighted gradients of previous clean data and newly cleaned data)

- **HoloClean**
  - Clean and enrich based on quality rules, value correlations, and reference data
  - Probabilistic models for capturing data generation
  - HoloDetect
    - Learn data representations of errors
    - Data augmentation w/ erroneous data from sample of clean data

- **Other Systems**
  - **AlphaClean** (generate data cleaning pipelines) [preprint]
  - **BoostClean** (generate repairs for domain value violations) [preprint]
Summary and Q&A

- Data Acquisition and Integration
- Data Preparation and Feature Engineering
- Data Transformation and Cleaning

Next Lectures
- 10 Model Selection and Management [Jun 05]
  - Incl Data Augmentation
- 11 Model Debugging Techniques [Jun 12]
- 12 Model Serving Systems and Techniques [Jun 19]

[Andreas C. Mueller: Preprocessing and Feature Transformations, Applied ML Lecture 2020]

“Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering”

– Andrew Ng