BASIC FEATURE MANIPULATION

• Transformations (Recap)
  • Centering
  • Min-Max scaling
  • Standardization

• Missing Values
  • impute with mean, median, or mode
WHAT IS FEATURE ENGINEERING?

• Create a *new set of features* given the training data.

• Why?
  • data exploration and visualization
  • increase performance of supervised learning
    → better predictions and/or
    → faster training/prediction time (efficiency)

• How?
  • *feature selection* (cf. Lab1 (Iris), Lab4 (Boston housing))
    • select subset of existing features
    • generates less features
  • *feature generation* (cf. hw4 → polynomial features)
    • create new features from existing features
    • generates less or more features
(STATISTICAL) FEATURE SELECTION

Use **statistical properties** of...

• ...features → remove *low variance* features

![](image)

• ...features and targets → remove features with *low relationship* to the target variable

  • correlation (*F-test*)
  • *mutual information*

\[
R(X, Y) = \frac{E[(X-\text{mean}(X))(Y-\text{mean}(Y))]}{\text{std}(X) \cdot \text{std}(Y)}
\]

\[
I(p(X), p(Y)) = D_{KL}(p(X,Y)||p(X)p(Y))
\]

we need some more **probability and statistics** knowledge to understand those
(ML-BASED) FEATURE SELECTION

Evaluate performance of a supervised ML model to find set of best features:

• **brute-force**: create datasets with all possible combinations of features → evaluate each of their performances → select best performing dataset

• **iterative/recursive feature elimination** → more efficient

```
RFE(X, y)
FOR j = 1, ..., size(X)
    perf[j] = CV(X \{x_j\}, y)
ENDF
worst_j = argmax(perf)
X_new = X \{x_{worst_j}\}
IF NOT stopping-criteria
    RFE(X_new, y)
ELSE
    RETURN X_new
```

Notes:
• **greedy procedure**
• requires a **validation set** or cross-validation
• can also be performed bottom-up (adding features)
• also called **wrapper/filter**
(ML-BASED) FEATURE SELECTION

Use a supervised ML model directly to find features:

- **model-based feature selection**
  - model computes *feature importance*
  - select most important features
  - not all ML models can compute feature importance
    - logistic regression: feature weight \( \times \text{std}(x) \)
    - random forest: *mean decrease impurity*

\[ \text{→ decrease in node impurity (weighted by the number of samples it splits) is summed and averaged across all trees} \]
ASIDE: QUICK INTRO TO DECISION TREES

- explainable, easy to interpret ML model

...continue splitting until all leaves are pure

select **best feature** to split on based on **entropy (information gain)** or resulting **node impurity**

math & statistics

\[
\text{Entropy}(S) = \sum_{v \in V(S)} \frac{|S_v|}{|S|} \log_2 \left( \frac{|S_v|}{|S|} \right)
\]

where $S$ is a collection of values
$V(S)$ is the set of unique values in $S$
$S_v$ is the collection of elements in $S$ with value $v$
ASIDE: QUICK INTRO TO DECISION TREES

- best feature to split on based on *node impurity*

  - Split on Gender
    - **Female**
      - Students = 10
      - Play Cricket = 2 (20%)
    - **Male**
      - Students = 20
      - Play Cricket = 13 (65%)

  - Split on Height
    - < 5.5 ft
      - Students = 12
      - Play Cricket = 5 (42%)
    - >= 5.5 ft
      - Students = 18
      - Play Cricket = 10 (56%)

  - Split on Class
    - **Class IX**
      - Students = 14
      - Play Cricket = 6 (43%)
    - **Class X**
      - Students = 16
      - Play Cricket = 9 (56%)

- do this for each node/each level
ASIDE: QUICK INTRO TO DECISION TREES

• 2D Example:
ASIDE: QUICK INTRO TO RANDOM FORESTS

- uses an *ensemble* of short decision trees
- most powerful, off-the-shelf ML method

![Diagram of Random Forests]

1. **X dataset**
2. **$N_1$ features**
   - **TREE #1**
   - **CLASS C**
3. **$N_2$ features**
   - **TREE #2**
   - **CLASS D**
4. **$N_3$ features**
   - **TREE #3**
   - **CLASS B**
5. **$N_4$ features**
   - **TREE #4**
   - **CLASS C**

**MAJORITY VOTING**

**FINAL CLASS**
FEATURE SELECTION - DISCUSSION

• open question: how many features to select?
  • based on desired efficiency
  • based on desired performance/prediction accuracy
    • may be learned similar to “model selection”
      – validation set
      – cross-validation
FEATURE GENERATION

• **use k-means**
  • compute clustering
  • use cluster membership as feature or
  • use distance to each cluster center as feature

• **principal component analysis (PCA)**
  • find directions of biggest spread
  • rotate data to new axis

• **neural networks/deep learning**

Also:

- feature learning
- representation learning

reduces the number of features

unsupervised

unsupervised or supervised
ASIDE: QUICK INTRO TO NEURAL NETS

• Recap: Linear Regression
• Example: selling my car

How did we solve this?
• choose loss function to measure error (RSS)
• minimize loss
  • derive the gradient
  • set it to zero
  • solve for w, b
ASIDE: QUICK INTRO TO NEURAL NETS

• Many layers of linear classifiers ("neurons")
  ➔ millions of slopes and intercepts

input data point

• iterative solution
  • start with random parameters $w$
  • derive the gradient of the loss wrt $w$
  • update parameters in the direction of descending gradient

• brute-force solution
  • Exhaustive parameter search

How to solve?
  • choose a loss function to measure error

output signal

DOG
CAT
TREE
CAR
SKY

14

ASIDE: QUICK INTRO TO NEURAL NETS

• What do the *neurons* mean?

Deep neural networks learn hierarchical feature representations.
FEATURE LEARNING VIA NNS

• use the NN weights (of the last full/hidden layer) as features

→ new features for this input (image) $\hat{\mathbf{z}} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ \vdots \\ Z_D \end{bmatrix}$
FEATURE LEARNING VIA NNS

• **problem**: training NNs is *extremely challenging*
  • needs a lot of training data (which we usually don’t have)
  • time-consuming
  • infinitely may model selection choices → number of layers, number of neurons in each layer, activation functions, etc.

• **solution**: use *pre-trained network* (cf. Lab9)
PRE-TRAINED NN FOR IMAGES

- VGG16

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories.

Other pretrained NNs:
FEATURE GENERATION - DISCUSSION

• using NNs we can turn *non-vectorial* input data (like images, text, etc.) into a vector representation

• more features is not always better
  • we still need to evaluate the new feature representations (via testing/cross-validation)

• the model used for feature generation (or selection) can/will be **different** from the model used for training/prediction
SUMMARY & READING

• Feature engineering
  • is as **important** as model selection
  • depends on the **domain/data set**
  • can be performed in a **supervised** or **unsupervised** way

• **Feature selection** *reduces* the number of features.

• **Feature generation** typically *increases* the number of features (not PCA).

• [DSFS]
  • Ch10 Dimensionality Reduction (p134-139)
  • Ch17 What is a Decision Tree? (p201-203), Random Forest (p211)

• [PDSH]
  • Ch5 Feature Engineering (p375-381)