Lecture-4-Feature-Extraction_short

April 24, 2019

1 Machine Learning

Lecture 4
Feature Extraction Part 1

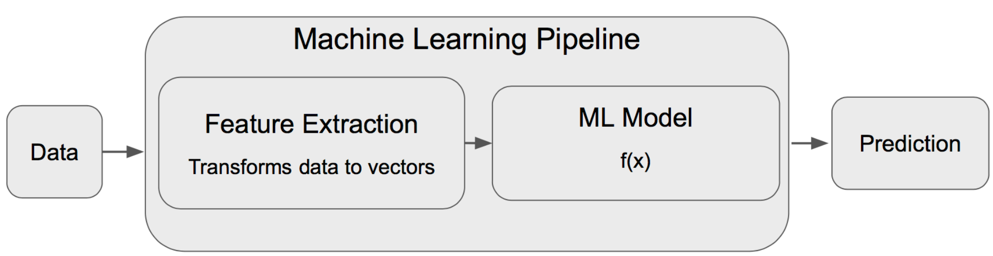
2 Machine Learning Pipelines

3 Feature Extraction

- Feature extraction describes the transformations from **data** to **vectors**
- Extracting good features is **more important** than choosing a ML model
- We will cover some standard feature extractors, but there are many more
- Often the best feature extractors include domain knowledge by human experts
- Feature extractors have to be optimized along with all other model parameters

4 Feature Extraction

- Continuous Features
- Categorical Features
- Text (next lecture)
- Images (next lecture)



ml-pipeline-2.png

4.1 Continuous Features

- Continuous features: $x \in R^d$
- For many models continuous features don't need to be transformed
- For some models it is necessary or beneficial to **normalize** continuous features
  - When optimizing with stochastic gradient descent
  - When regularizing models: to control regularization

4.2 Normalization: z-scoring

Given a feature $x \in R^1$ (and for multivariate x analogously) there are several standard normalization options:

Standard scaling / z-scoring: compute mean μ_x and standard deviation σ_x of x and compute

$$x \leftarrow \frac{(x - \mu_x)}{\sigma_x}$$

- Resulting variable has zero mean and unit variance
- See sklearn.preprocessing.StandardScaler

```
In [1]: from sklearn.preprocessing import StandardScaler
        data = [[0, 0], [0, 0], [1, 1], [1, 1]]
        scaler = StandardScaler()
        scaler.fit(data)
        print("Estimated Mean: {}".format(scaler.mean_))
        scaler.transform([[.5, .5], [2,2]])
```

```
Estimated Mean: [0.5 0.5]
```

```
Out[1]: array([[0., 0.],
               [3., 3.]])
```

4.3 Normalization: Min-Max Scaling

Min-max scaling: compute min $\min_x$ and max $\max_x$ of x

$$x \leftarrow \frac{(x - \min_x)}{\max_x - \min_x}$$

- Resulting variable is in range [0,1] (or any other range)
- See sklearn.preprocessing.MinMaxScaler

4.4 Categorical Features

Categorical features are variables $x \in \{0, 1, 2, \ldots, N\}$ taking one value of a set of cardinality N without an implicit ordering e.g.: - colors (red, green, blue) - user ids - Movie ids

Often used feature transformations are

- One-hot encoding
- More recently for neural networks: low dimensional embeddings

4.5 One-hot encoding

Given a set of values [red, green, blue], we transform the data into

- red $\leftarrow [1,0,0]$
- green $\leftarrow [0,1,0]$
- blue $\leftarrow [0,0,1]$

Sets of categorical variables can be represented as sum over single value vectors. Examples: bag-of-words vectors

```
In [2]: from sklearn.preprocessing import OneHotEncoder
        enc = OneHotEncoder()
        X = [['red'], ['green'], ['blue'], ['red']]
        enc.fit(X)
        enc.transform([['red'], ['green'], ['blue']]).toarray()
```

```
Out[2]: array([[0., 0., 1.],
               [0., 1., 0.],
               [1., 0., 0.]])
```

4.6 One-hot encoding: Problems

- Cardinality needs to be estimated upfront
- New items / categories cannot be represented
- Similarity information is lost (light-blue and blue as different as black and white)