Feature Engineering in Machine Learning

Zdeněk Žabokrtský
Institute of Formal and Applied Linguistics,
Charles University in Prague

Used resources

- http://www.cs.princeton.edu/courses/archive/spring10/cos424/slides/18-feat.pdf
- http://stackoverflow.com/questions/2674430/how-to-engineer-features-for-machine-learning
- https://facwiki.cs.byu.edu/cs479/index.php/Feature_engineering
- documentation of scikit-learn
- wikipedia

Human's role when applying Machine Learning

- Machine learning provides you with extremely powerful tools for decision making ...
- ... but until a breakthrough in AI, the role of the developer's decision will still be crucial.

Your responsibility:

- setting up the correct problem to be optimized (it's far from straightforward in the real world)
- choosing a model
- choosing a learning algorithm (or a family of algorithms)
- finding relevant data
- designing features, feature representation, feature selection ...

Feature

a feature - a piece of information that is potentially useful for prediction

Feature engineering

- feature engineering - not a formally defined term, just a vaguely agreed space of tasks related to designing feature sets for ML applications
- two components:
  - first, understanding the properties of the task you're trying to solve and how they might interact with the strengths and limitations of the model you are going to use
  - second, experimental work were you test your expectations and find out what actually works and what doesn't.

Feature engineering in real life

Typically a cycle

1. design a set of features
2. run an experiment and analyze the results on a validation dataset
3. change the feature set
4. go to step 1

Don't expect any elegant answers today.

Causes of feature explosion

- **Feature templates:** When designing a feature set, you usually quickly turn from coding individual features (such as 'this word is predeced by a preposition and a determiner') to implementing feature templates (such as 'the two preceding POSs are X a Y')
- **Feature combination:** linear models cannot handle some dependencies between features (e.g. XOR with binary operations, polynomial dependencies with real-valued features) - feature combinations might work better.
- Both lead to quick growth of the number of features.

Stop the explosion

There must be some limits, because

- Given the limited size of training data, the number of features that can be efficiently used is hardly unbounded (overfitting).
- Sooner or later speed becomes a problem.

Possible solutions to avoid the explosion

- feature selection
- regularization
- kernels

Feature selection

- Central assumption: we can identify features that are redundant or irrelevant.
- Let's just use the best-working subset: $\arg\max_f acc(f)$, where $acc(f)$ evaluates prediction quality on held-out data
- Rings a bell? Yes, there's a set-of-all-subsets problem (NP hard), exhaustive search clearly intractable.
- (The former implicitly used e.g. in top-down-induced of decision trees.)
- A side effect of feature reduction: improved model interpretability.

Feature selection

Basic approaches:

- wrapper - search through the space of subsets, train a model for current subset, evaluate it on held-out data, iterate... simple greedy search heuristics:
  - forward selection - start with an empty set, gradually add the "strongest" features
  - backward selection - start with the full set, gradually remove the "weakest" features

  computationally expensive
- filter - use N most promissing features according to ranking resulting from a proxy measure, e.g. from
  - mutual information
  - Pearson correlation coefficient
- embedded methods - feature selection is a part of model construction

Regularisation

- regularisation = introducing penalty for complexity
- the more features matter in the model, the bigger complexity
- in other words, try to concentrate the weight mass, don't scatter it too much
- application of Occam's razor: the model should be simple
- Bayesian view: regularization = imposing this prior knowledge ("the world is simple") on parameters

Regularisation

In practice, regularisation is enforced by adding a factor that has high values for complex parameter settings to the cost function, typically to negative log-likelihood:

$$cost(f) = -l(f) + regularizer(f)$$

- L_0 norm: $... + \lambda count(w_j \neq 0)$ - minimize the number of features with non-zero weight, the less the better
- L_1 norm: $... + \lambda |w|$ - minimize the sum of all weights
- L_2 norm: $... + \lambda ||w||$ - minimize the lenght of the weight vector
- $L_{1/2}$ norm: $... + \lambda \sqrt{||w||}$
- L_∞

Experimenting with features in scikit-learn

Encoding categorical features

Turning (a dictionary of) categorical features to a fixed-lenght vector:

- estimators can be fed only with numbers, not with strings
- turn a categorical feature to one-of-K vector of features
- `preprocessing.OneHotEncoder` for one feature after another
- or `sklearn.feature_extraction.DictVectorizer` for the whole dataset at once, two steps: `fit_transform` and `transform`

Feature binarization

- thresholding numerical features to get boolean values
- `preprocessing.Binarizer`

Feature Discretization

- converting continuous features to discrete features
- Typically data is discretized into partitions of K equal lengths/width (equal intervals) or K% of the total data (equal frequencies).
- ? in sklearn?

Dataset standardization

- some estimators might work badly if distributions of values of different features are radically different (e.g. in the order of magnitude)
- solution: transform the data by moving the center (toward zero mean) and scaling (towards unit variance)
- `preprocessing.scale`

Vector normalization

- Normalization is the process of scaling individual samples to have unit norm.
- solution: transform the data by moving the center (toward zero mean) and scaling (towards unit variance)
- `preprocessing.normalize`

Feature selection

Scikit-learn exposes feature selection routines as objects that implement the transform method:

- `SelectKBest` removes all but the k highest scoring features
- `SelectPercentile` removes all but a user-specified highest scoring percentile of features
- using common univariate statistical tests for each feature: false positive rate SelectFpr, false discovery rate SelectFdr, or family wise error SelectFwe.

These objects take as input a scoring function that returns univariate p-values:

- For regression: `f_regression`
- For classification: `chi2` or `f_classif`