Features: representation, normalization, selection

Chapter 10

Features

- Distinguish between instances (e.g., an image that you need to classify), and the features you create for an instance.
- Features are "the workhorses of machine learning": the quality of a classifier is crucially dependent on the features used to represent the domain.
- Although many datasets come with pre-defined features, they can be manipulated in many ways:
  - Normalization or other transformations
  - Discretization
  - Select the best features
  - Combine features to compute new ones (feature construction)

Types of features

Quantitative (continuous) features: have a meaningful numerical scale
- Examples: age, height etc.

Ordinal features: have an ordering
- Example: house number

Categorical (discrete/nominal) features: not meaningful to describe them using mean, median
- Example: Boolean features
Features and classifiers

Some classifiers require numerical quantitative (linear classifiers).

Need to transform an ordinal/categorical feature into a numerical feature.

Assume we have a feature with possible values "lo", "med", "hi". How would we transform this into one or more quantitative features?

Note: SVMs and other kernel-based methods can be applied to ordinal/categorical data by way of an appropriate kernel function.

Features and classifiers

Some classifiers treat categorical and quantitative features differently.

Examples: Naïve Bayes, decision trees

Sometimes useful to use categorical features even if the underlying data is quantitative.
Solution: discretization

Aside: how about nearest neighbor classifiers?

Feature transformations

Aim: improve the utility of a feature by changing it in some way.

Feature transformations

Discretization: a mapping from a continuous domain to a discrete one.

Thresholding: Simplest form of discretization that constructs Boolean features.

How to choose the threshold?

- Unsupervised: mean/median
- Supervised: choosing the threshold to make the feature most predictive (need a measure of how predictive a feature is)
Thresholding a feature

Coverage curve obtained by ranking countries on decreasing GDP per capita, using 23 Euro countries as the positive class. The orange split sets the threshold equal to the mean and the green split sets the threshold equal to the median. The red points are on the convex hull of the coverage curve and indicate potentially optimal splits when the class label is taken into account.

(left) Coverage curve of the same feature, using 50 countries in the Americas as the positive class.

Discretization

Things to consider:
- How many values to use
- Thresholds.

Unsupervised methods:
- Choose bins such that each value has approximately the same number of instances
- Equal-width discretization
- Discretization via clustering

Algorithm 10.1: Recursive partitioning

\[
\text{RecPart}(S, f, Q) \]

Input: set of labelled instances S ranked on feature values \( f(x) \); scoring function \( Q \).

Output: sequence of thresholds \( t_1, \ldots, t_{k-1} \).

1. If stopping criterion applies then return \( \emptyset \);
2. Split \( S \) into \( S_L \) and \( S_R \) using threshold \( t \) that optimises \( Q \);
3. \( T_L = \text{RecPart}(S_L, f, Q) \);
4. \( T_R = \text{RecPart}(S_R, f, Q) \);
5. return \( T_L \cup \{t\} \cup T_R \);

Normalization

Feature normalization: neutralize the effect of different scales across features (geometric classifiers are sensitive to that).
- Standardization
- Scaling to [0,1]

Instance normalization: normalize a feature vector to have unit norm. Appropriate for sparse features.

The two forms of normalization can be combined.
Logistic feature calibration

(left) Two-class Gaussian data. The middle line is the decision boundary learned by the basic linear classifier; the other two are parallel lines through the class means.

Logistic calibration to probability space is a non-linear transformation that pushes data away from the decision boundary.

Figure 10.5

Missing data

In many cases an instance may be incomplete, i.e. some feature values will be missing.

Decision tree models and some probabilistic models can handle this well.

Geometric models (SVM, KNN) need all features to be specified.

Solution: fill in missing values (imputation).

Simple solution: use the mean

More sophisticated: build a predictive model

Complications: the fact that the feature is missing might be correlated with the label

Feature selection and construction

Distinguish between

Feature selection: select a subset of features that allow a classifier to maintain or increase its performance.

Feature construction: construct new features that are a linear/non-linear combination of the original features.

Feature selection

Why do feature selection?
Feature selection

Why do feature selection?
- Better understanding of the data and the classification rule.
- Expensive to compute/measure all features
- Improve classifier performance: some machine learning algorithms, are known to degrade in performance when faced with many irrelevant/noisy features

Objective: find a subset of the original features, such that a classifier that is run on data containing only these features generates a classifier with the highest possible accuracy.

In practice feature selection can lead to improved performance. This depends on the classifier used: this is definitely the case for KNN or naive Bayes classifiers, but not necessarily the case for SVMs.

Comment: Relevance does not imply that a feature should be in an optimal feature subset.

Approaches

Filter: Features or groups of features are scored by some measure of correlation with the labels. Examples of scoring functions include Pearson correlation between a feature and the labels, Fisher scores, or mutual information.

Wrapper: Uses the classifier to guide the process of selection. A feature subset is typically evaluated using CV.

Embedded: The selection method uses properties of the classifier.

Filter methods

- Pearson correlation of a feature with the labels
- Difference in a feature's mean across classes: the Golub score:
  \[
  \frac{|\mu_{\oplus} - \mu_{\ominus}|}{\sigma_{\oplus} + \sigma_{\ominus}}
  \]
How many features?

Drawbacks of filter methods?

How to decide on how many features?
- Decide ahead of time.
- Use CV.

Wrapper methods

Iteratively add features (forward selection) or eliminate them (backward elimination)

Use cross validation to guide feature inclusion/removal:
Rank features by how much they add to accuracy, or by how much accuracy decreases by their removal.

This is a greedy approach.
How many runs of cross-validation do we need to run?

Embedded methods

We have seen that the magnitude of the weight vector of a linear classifier can be used as a measure of feature importance.

Recursive Feature Elimination (RFE):
Alternate between training an SVM and removing the feature with the lowest magnitude of the weight vector.

For high dimensional datasets you can remove a fraction of the features at each iteration.

Which type of methods should I use?

The wrapper and embedded approaches take into account the heuristics and biases of the classifier that will ultimately be used, and thus can potentially generate better feature subsets. Filter methods are less likely to suffer from overfitting.

So ultimately, it depends on the data.
Bias in feature selection

The wrong way to evaluate feature selection:

Perform feature selection on the whole dataset and then run cross-validation.

Why is this wrong?

Bias in feature selection

The wrong way to evaluate feature selection:

Perform feature selection on the whole dataset and then run cross-validation.

The right way:

Perform feature selection using training data only.

Feature selection in PyML

Feature selection methods:
- RFE
- Filter methods

Support proper experiment design

Show a demo!