AIML430/COMP309: ML Tools and Techniques
Lecture 11: Dimensionality Reduction Methods

Ali Knott
School of Engineering and Computer Science, VUW

The plan for today and next week

Today (and next week) we're looking at 'feature manipulation' methods.

Today: a more detailed look at *dimensionality reduction*.

- *Global methods* for changing feature space (PCA, & new ones)
  - Sometimes called feature extraction.
- *Local methods* for removing features
  - Sometimes called feature selection.

Next week (with Marcus Frean): 'Feature engineering'.

- In particular: how to *construct* useful new features.
  - Discretisation (yesterday) is one way to do this...
  - Marcus will show you others!

Why do feature manipulation?

Raw data contains all sorts of features...

- Not all necessarily useful for your purposes.

Some reasons for manipulating features:

- To improve performance of the model to be learned.
  - Improvements could be in speed, predictive performance, simplicity, interpretability...
- To reduce noise. (And to reduce overfitting.)
- To compress the data.
- To visualize something in the data.

1. Feature extraction

Feature extraction is a *global* transformation of the feature space.

- We start with a space of d dimensions, and transform to m dimensions. (m normally smaller than d.)
- The idea: to retain most of the relevant information in the original feature space.

Feature extraction

There are two main approaches to feature extraction.

One is projection: project points into a lower-dimensional space.

- To visualise projection: think of points ‘falling onto’ a straight line, or plane.

The other is manifold learning.

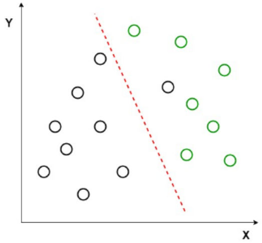
- A manifold is a topological space that maps to your feature space.
- The manifold can be *curved*—so we can model complex patterns.
- But at any *local* point, its topology *resembles* that of the feature space.
  - Your data points might live on a *sphere*, or a *saddle*, defined within your feature space…

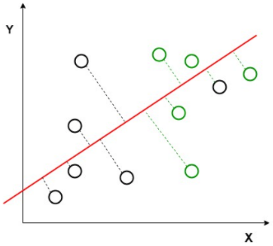
1.1 Projection techniques for feature extraction

We have already seen two projection techniques: principal component analysis (PCA) and independent components analysis (ICA).

Another one is linear discriminant analysis (LDA).

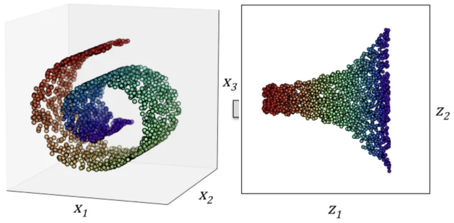
- This method simplifies *input space* for a classification task.
- It finds linear decision boundaries in feature space, and projects data onto planes perpendicular to those boundaries.
- sklearn.discriminant_analysis.LinearDiscriminantAnalysis





1.2 Manifold techniques for feature extraction

A useful visualisation of reducing dimensionality with a manifold:



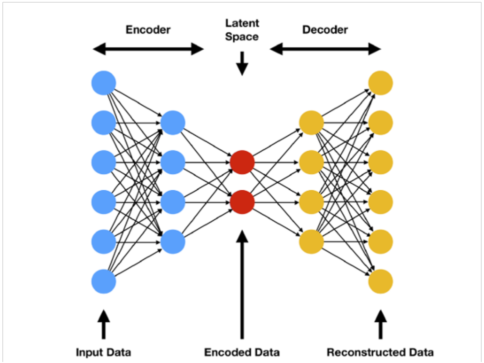
We already mentioned t-sne.

- The idea here is to transform feature space into an embedding space. . .
  - With fewer dimensions. . .
  - Which is *optimised*, to preserve the pairwise distances between all datapoints.
- Optimisation happens iteratively, using gradient descent.

1.2 Manifold techniques for feature extraction

We've also already seen autoencoders.

- This is a multi-layer perceptron that learns to map each datapoint *back onto itself*. . .
- Through a 'bottleneck layer' *smaller* than the input/output layers.
- 'Embedding' is a ML term used for any learned representation.



2. Feature selection techniques

The idea here is to *remove individual features* from the original dataset.

- Find uninformative or redundant features, and get rid of them!

There are two types of FS technique.

- Univariate methods consider each feature by itself.
  - Mostly: how much does it *covary* with other features?
- Multivariate methods consider all features at once.
  - E.g. in algorithms that iterate through each feature.

2.1. Univariate feature selection methods

Univariate methods run *tests* on each input feature, to see how it relates to other features.

- Strong relationships with the *output feature* are good!
- Strong relationships with *other input features* are bad!

Based on the results of these tests, you can em select features.

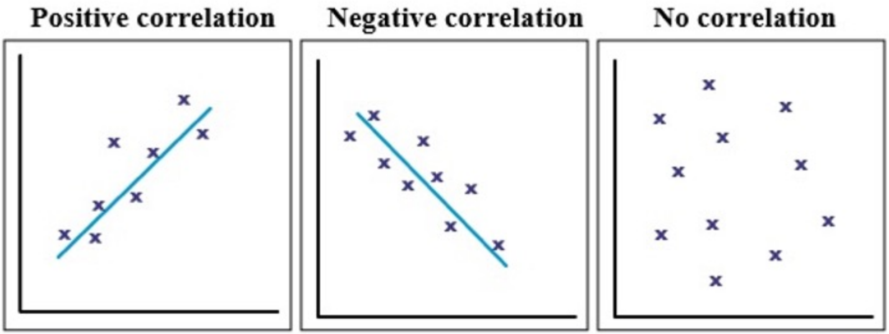
- Keep the good ones, discard the bad ones!

We'll review some of the tests you can use.

- All in sklearn's feature_selection module.
- Key functions: SelectKBest, SelectPercentile

2.1. Univariate feature selection methods

For a *regression* task, you should use tests based on *linear correlation*.



Use sklearn.feature_selection.r_regression()

2.1. Univariate feature selection methods

For a *classification* task, you should use tests based on a *discrete* output variable.

- A good test is chi-squared: this test identifies features that are likely to be independent of the output class.
- It tests the (null) hypothesis that there's no association between a feature and the output feature.

Use sklearn.feature_selection.chi2()

Sklearn offers several other univariate tests. . .

There are methods based on mutual information calculations.

- Mutual information measures the *reduction in uncertainty* about Variable 1, that comes from knowing the value of Variable 2.
  - For regression tasks, use sklearn.feature_selection.mutual_info_classif
  - For classification, use sklearn.feature_selection.mutual_info_regression

For continuous variables that aren't normally distributed, Spearman's rank correlation coefficient is a good measure of correlation.

- Use scipy.stats.spearmanr

For continuous inputs and discrete outputs, Analysis of Variance (ANOVA) is also useful. (Use sklearn.feature_selection.f_classif)

- This finds the amount of variance in the output class that's *explained* by the input variable. (Expressed as a proportion.)

2.1. Univariate model-based feature selection

Another way of identifying (individual) features is to train a model, and then *analyse the model*.

- This is done with sklearn.feature_selection.SelectFromModel.
- With a *regression* model, for instance, scores can come straight from *coefficients* for input variables.
- With a *decision tree*, there are higher scores for the features used high up in the tree.

2.2. Multivariate feature selection methods

Selecting individual features is not always ideal, because sometimes features *interact*: so their importance can't be evaluated in isolation.

- Multivariate feature selection methods assess the importance of *groups of features*.

Multivariate methods are often implemented as 'wrappers' around model training processes.

- The basic idea is to run *many* training processes, experimenting with different features–and see which feature set does best.
- Obviously, it's *costly* to iterate many times over the whole training process!

2.2. Multivariate feature selection methods

There are two ‘standard’ wrapper methods for feature selection.

Sequential Forward Generation (SFG) starts with an empty set of ‘used features’: all features are ‘candidates’. At each iteration:

- For each candidate feature *C*, train a model that adds *C* to the set of used features. Pick the best-performing model.

This works best when the optimal feature set is small. (But it’s greedy!)

Sequential Backward Generation (SBG) starts with a full set of ‘used features’. At each iteration:

- For each used feature *U*, train a model *without U*. Pick the model whose performance decreases least.

This works best when the optimal feature set is big. (It’s also greedy!)

2.2. Multivariate feature selection methods

Advanced feature selection methods take less ‘greedy’ approaches to selecting/rejecting features.

- Essentially, we’re searching the space of *possible feature combinations*.
- If there are n features, there are 2^n possible combinations... How can we effectively explore that space?

For large feature sets, *exhaustive* search is *impossible*.

- We have to find good *heuristic* search methods.
- Our school specialises in this work!

A common heuristic search method is with genetic algorithms.

Genetic algorithms: A tiny taster

Imagine a population of ‘creatures’ that have *feature sets instead of genes*.

- They compete with each other, and only the fittest survive. . .
- Fitness is determined by how well their model performs!
- The survivors are randomly paired together, to ‘breed’ a *new generation* of creatures.

If you fancy this kind of synthetic biology, Vic’s AI group is for you! :-)

Summary: ‘Dimensionality reduction’ methods

Global methods for dimensionality reduction map data into *new feature spaces*.

- Projection methods (PCA, ICA, LDA) find good *linear* mappings…
- Manifold methods (t-SNE, audoencoders) learn good *nonlinear* mappings.

Local methods for dimensionality reduction *select* features from the full feature set.

- Univariate methods score *individual* features, & keep the best…
- Multivariate (‘wrapper-based’) methods assess *feature combinations*.
  - Simple methods (forward and backward) perform ‘greedy’ searches of feature combinations.
  - ‘Heuristic’ searches (e.g. genetic algorithms) are smarter—but also very expensive.

. . . And now over to Marcus Frean!

Next week, Marcus will take over.

The first thing he'll look at will be 'feature engineering': that is, how to *construct new features*.

- Discretisation is a simple way to do that. . . he will show you others!