

Intro. to machine learning (CSI 5325)

Lecture 19: Instance-based learning

Greg Hamerly

Some content from Tom Mitchell.

1 Advanced topics in instance-based learning

Advanced dimensionality reduction

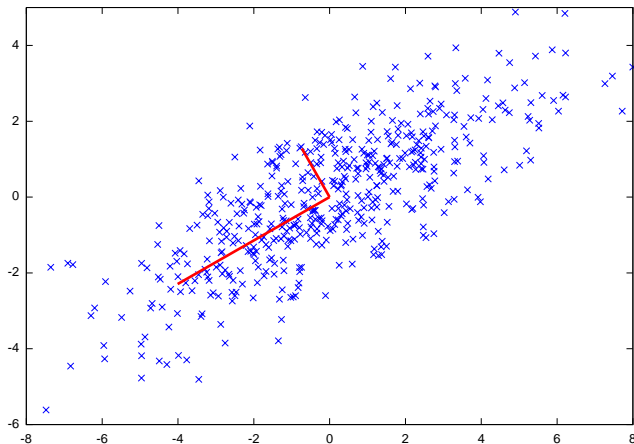
Basic premises:

- some attributes are irrelevant; others are noisy
- true signal often lives in a subspace which doesn't use all attributes (feature reduction), or uses a novel combination of them (feature generation)

Some more advanced or recent techniques:

- principal components analysis, Mahalanobis distance
- random linear projections
- nonlinear manifold learning
- multidimensional scaling
- alternative distance metrics (L_1 , etc.)
- distance metric learning

Principal components analysis



PCA (2)

PCA is a linear transformation of the data into a new basis.

It assumes that there is some orthogonal vector basis of the original data which captures the variance of the data in a better way.

In the transformed data, the first dimension has the largest variance, the second dimension has the second largest variance, etc.

It finds a new set of orthogonal basis vectors in the original space that point in the directions of maximum variance.

PCA (3)

Based on finding the eigenvalues of the covariance of the data:

```
[v, lambda] = eig(cov(x));
```

Matlab makes it even easier than this:

```
[v, z, lambda] = princomp(x);
```

If we find the principal components and use only the top few, we have reduced the dimension of our data while keeping the majority of the information (aka variance).

Random linear projections

Basic idea:

- choose k random vectors in the original space, where $k \ll d$
- linearly project all the data to the k new vectors
- learn in this lower-dimensional space

Even though it sounds crazy, amazing properties.

Johnson-Lindenstrauss lemma: if we project n points into $k = O(\log n)$ dimensions, then there is a high probability that the $O(n^2)$ inter-point distances will remain nearly the same.

- note – no dependence on the original d !

Application of Random linear projections

Random linear projections are a key part of my research in SimPoint, which uses machine learning to analyze program execution.

To begin with, we have data from an executed program:

- each example is a window of (e.g.) 1,000,000 instructions executed
- the attributes are the regions of code that execute
- the attribute values are the number of times each region executes during that time window

Application of Random linear projections (2)

The number of attributes is large: sometimes $> 100,000$ dimensions.

We use random linear projection to bring the dimension down to about 15 dimensions, which is much more workable.

As a result, our analyses run MUCH faster, and are nearly as good.

Multidimensional scaling

Given a set of pairwise distances (not original points), find a set of vectors which preserve those distances.

Given: $d(x_i, x_j)$, pairwise distances for n points ($1 \leq i, j \leq n$).

Find the minimum of

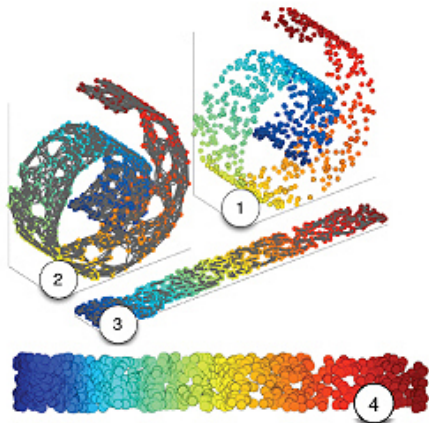
$$S(y_1, y_2, \dots, y_n) = \sum_{i \neq j} (\|y_i - y_j\| - d(x_i, x_j))^2$$

- where y_i are vectors in some space of some chosen dimension.
- want sufficient dimension for y_i , but not too high!
- method: minimize S by gradient descent on y_i vectors.

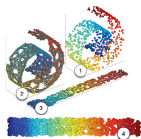
Nonlinear manifold learning

Basic idea: what if the data lives in some non-linear subspace?

The 'swiss roll':



Nonlinear manifold learning (2)



The 'swiss roll':

Common approaches (ISOMAP, etc.):

- construct a nearest-neighbor graph of all n points
- use this graph to estimate pairwise distances (e.g. using shortest path length)
- use MDS to obtain a new representation in a vector space

Alternative distance metrics (L_1 , etc.)

Minkowski (L_p) norm:

$$L_p(x) = \left(\sum_{i=1}^d |x_i|^p \right)^{1/p}$$

$$d_p(x, y) = \|x - y\|_p = L_p(x - y)$$

- L_2 = Euclidean distance
- L_1 = Manhattan distance
- L_∞ = max

General observation: in high dimension, lower p norms tend to work better (Aggarwal 2001).

Mahalanobis distance

Given a scaling matrix $A \in \mathcal{R}^{d \times d}$, the Mahalanobis distance between vectors $x, y \in \mathcal{R}^d$ is

$$\|x - y\|_A = \sqrt{(x - y)^T A (x - y)}$$

This has relevance with:

- Gaussian probability distribution ($A = \Sigma^{-1}$, where Σ is the covariance matrix)
- principal components analysis
- metric learning

Distance metric learning

For a set of $\{+, -\}$ labeled points, learn a metric that gives good nearest-neighbor classification accuracy.

Or, the input could be a set of labels of *pairs* of points as similar/dissimilar to adjust the metric for clustering (Xing et al. 2002).

More general than scaling of individual coordinates: learn A in the Mahalanobis distance.