basic concepts

Thursday, May 18, 2017   3:23 PM

In this book, we will be focussing on probabilistic models of the form $p(y|\mathbf{x})$ or $p(\mathbf{x})$, depending on whether we are interested in supervised or unsupervised learning respectively. There are

why no conditioanlity?

- in supervised after we build model we want to be able to feed data to it and get answers (class/value)
- in unsupervised we are interested in grouping of the input data; I don't think we can reuse such model to ask about new points; but we can use the results to build a new model?

parametric vs non-parametric

have a fixed number of parameters, or does the number of parameters grow with the amount of training data? The former is called a **parametric model**, and the latter is called a **non-parametric model**. Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions. Non-parametric models are more flexible, but often computationally intractable for large datasets.

examples

A simple example of a non-parametric classifier is the K **nearest neighbor** (**KNN**) classifier. This simply "looks at" the K points in the training set that are nearest to the test input $\mathbf{x}$,

the curse of dimensionality

There are multiple phenomena referred to by this name in domains such as numerical analysis, sampling, combinatorics, machine learning, data mining, and databases. The common theme of these problems is that **when the dimensionality increases, the volume of the space increases so fast that the available data become sparse**.
This **sparsity is problematic** for any method that requires statistical significance. In order to obtain a statistically sound and **reliable result, the amount of data needed to support the result often grows exponentially** with the dimensionality. Also, organizing and searching data often relies on detecting areas where **objects form groups with similar properties**; in high dimensional data, however, **all objects appear to be sparse and dissimilar in many ways**, which prevents common data organization strategies from being efficient.

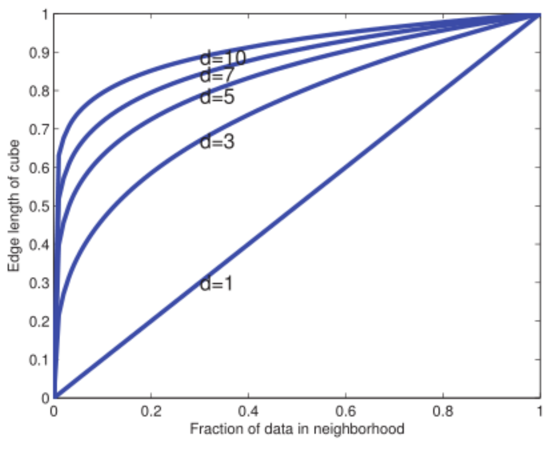
From <https://en.wikipedia.org/wiki/Curse_of_dimensionality>

in KNN classifier

The KNN classifier is simple and can work quite well, provided it is given a good distance metric and has enough labeled training data. In fact, it can be shown that the KNN classifier can come within a factor of 2 of the best possible performance if $N \to \infty$ (Cover and Hart 1967).

However, the main problem with KNN classifiers is that they do not work well with high dimensional inputs. The poor performance in high dimensional settings is due to the **curse of dimensionality**.

To explain the curse, we give some examples from (Hastie et al. 2009, p22). Consider applying a KNN classifier to data where the inputs are uniformly distributed in the D-dimensional unit cube. Suppose we estimate the density of class labels around a test point $\mathbf{x}$ by "growing" a hyper-cube around $\mathbf{x}$ until it contains a desired fraction f of the data points. The expected edge length of this cube will be $e_D(f) = f^{1/D}$. If $D = 10$, and we want to base our estimate on 10% of the data, we have $e_{10}(0.1) = 0.8$, so we need to extend the cube 80% along each dimension around $\mathbf{x}$. Even if we only use 1% of the data, we find $e_{10}(0.01) = 0.63$: see Figure 1.16. Since the entire range of the data is only 1 along each dimension, we see that the method is no longer very local, despite the name "nearest neighbor". The trouble with looking at neighbors that are so far away is that they may not be good predictors about the behavior of the input-output function at a given point.



(b)

in sampling

There is an exponential increase in volume associated with adding extra dimensions to a mathematical space. For example, 10^2=100 evenly spaced sample points suffice to sample a unit interval (a "1-dimensional cube") with no more than 10^{-2}=0.01 distance between points; an equivalent sampling of a 10-dimensional unit hypercube with a lattice that has a spacing of 10^{-2}=0.01 between adjacent points would require 10^{20}[=$(10^2)^{10}$] sample points. In general, with a spacing distance of 10^{-n} the 10-dimensional hypercube appears to be a factor of $10^{n(10-1)}$[=$(10^n)^{10}/(10^n)$] "larger" than the 1-dimensional hypercube, which is the unit interval.

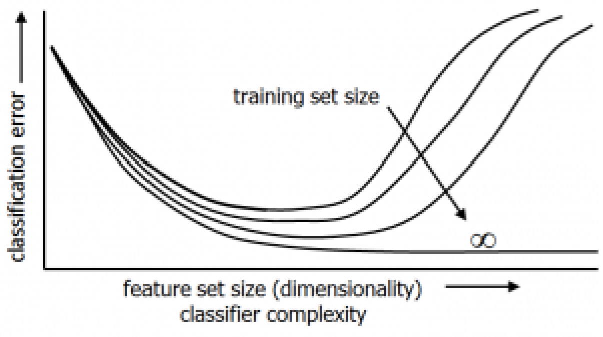
From <https://en.wikipedia.org/wiki/Curse_of_dimensionality>

in distance functions

When a measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples.
One way to illustrate the "vastness" of high-dimensional Euclidean space is to compare the proportion of an inscribed hypersphere to that of a hypercube. As the dimension of the space increases, the hypersphere becomes an insignificant volume relative to that of the hypercube.

From <https://en.wikipedia.org/wiki/Curse_of_dimensionality>

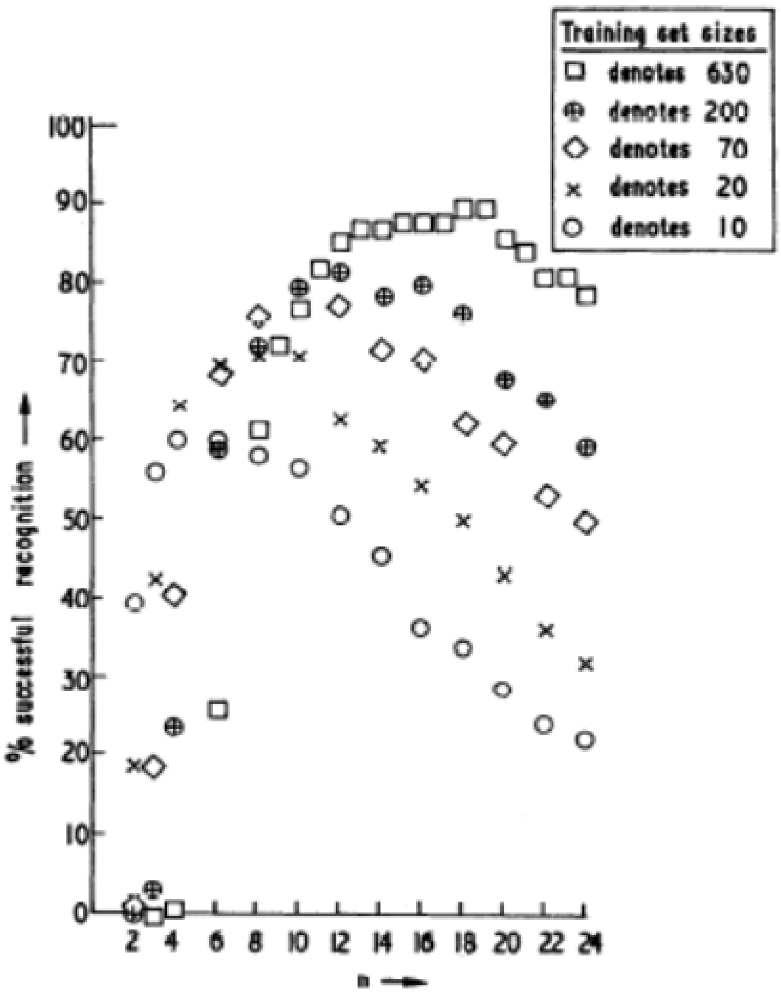
the peaking paradox



fundamental understanding in science: **if we measure more, we know more**
problem:
Designers of pattern recognition systems frequently encounter plots as above. They measure more features in order to describe the objects in finer details. This approach may initially help but at some moment it becomes counterproductive. More features increase the complexity of the recognition system. This introduces more parameters to be estimated. The corresponding estimation errors destroy the potentially accessible classification performance. These errors are caused by the limited size of the training set. The solution is

thereby to increase this set as well. Only for an infinite training set size holds that the performance is a monotonic function of the number of observations.

From <http://37steps.com/2279/the-peaking-paradox/>

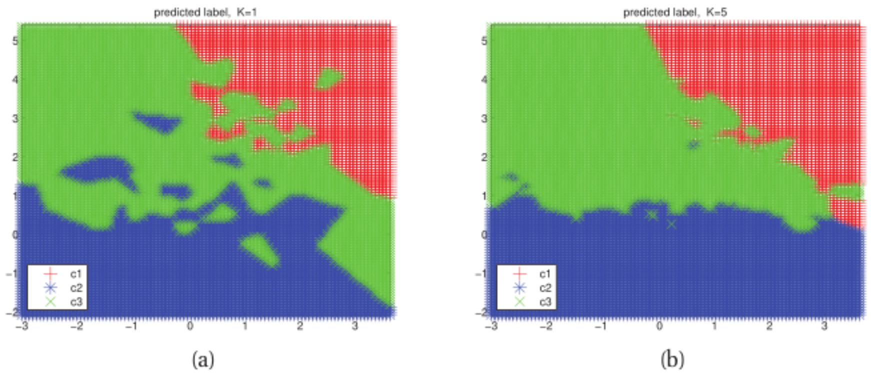


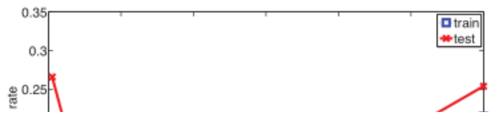
before it becomes counterproductive it can help

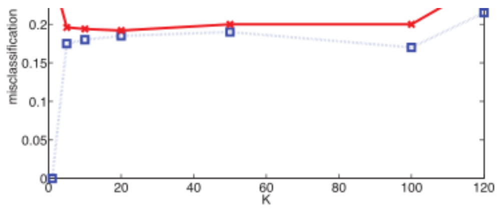
- scores are higher
- but more features are required to achieve them

overfitting

When we fit highly flexible models, we need to be careful that we do not **overfit** the data, that is, we should avoid trying to model every minor variation in the input, since this is more likely to be noise than true signal. This is illustrated in Figure 1.18(b), where we see that using a high







(a)

Figure 1.21 (a) Misclassification rate vs K in a K-nearest neighbor classifier. On the left, where K is small, the model is complex and hence we overfit. On the right, where K is large, the model is simple and we underfit. Dotted blue line: training set (size 200). Solid red line: test set (size 500). (b) Schematic

(don't pay attention to dotted blue line)

- small K gives us *complex* model - we get get a lot of cells
- large K gives us *simpler* model - far less cells possible

complex model is compuationally cheap, while simple is expensive in this case

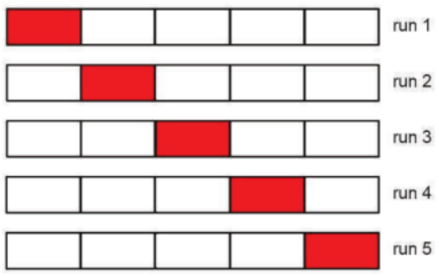
comparing models

misclassicfication rate - we can compare very different models

- we want generalizability, so need test and train sets (+ probably K-fold cross validation)

precision, recall and so on

K-fold crossvalidation



to get final result we average over all runs
each point used once for testing and K-1 times for learning