Machine Learning Course - CS-433

Curse of Dimensionality and k-NN

Oct 27, 2016

changes by Rüdiger Urbanke 2016

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Classification example

In many cases, a linear model may not be optimal. There are three contributing factors: perhaps our model is too rigid (bias), or perhaps it is too flexible (variance), or perhaps some errors are just unavoidable (the noise).

FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE*.*

*All figures taken from Chapter 2 HTF

The k-nearest-neighbor classifier/regressor is an entirely different way of performing classification/regression. It performs best if we are working in low dimensions and if we have reason to believe that points $(y, \mathbf{x})$ that are "close" in input have similar labels/values.

k-Nearest Neighbor (k-NN)

Assume that S_t is our training data. We are given a "fresh" input $\mathbf{x}$ and want to make a prediction. A natural k-NN prediction for $\mathbf{x}$ is,

$$f_{S_t,k}(\mathbf{x}) = \frac{1}{k} \sum_{n:\mathbf{x}_n \in \text{nbh}_{S_t,k}(\mathbf{x})} y_n \ ,$$

where $\text{nbh}_{S_t,k}(\mathbf{x})$ is the set of k input points in S_t that are closest to $\mathbf{x}$. There are many possible variants. E.g., instead of taking the empirical average we could weigh individual samples the heavier the closer they are to $\mathbf{x}$.
For the classification problem it is natural to output that label that appears the most frequently,

$$f_{S_t,k}(\mathbf{x}) = \text{majority element}\{y_n : \mathbf{x}_n \in \text{nbh}_{S_t,k}(\mathbf{x})\}.$$

For the binary case it is good to pick k to be odd so that there is a clear winner. For the simplest case of $k = 1$ we return the label of the closest neighbor.
Figures "2.3" and "2.2" show this rule applied to a binary classification problem with $k = 1$ an $k = 15$.

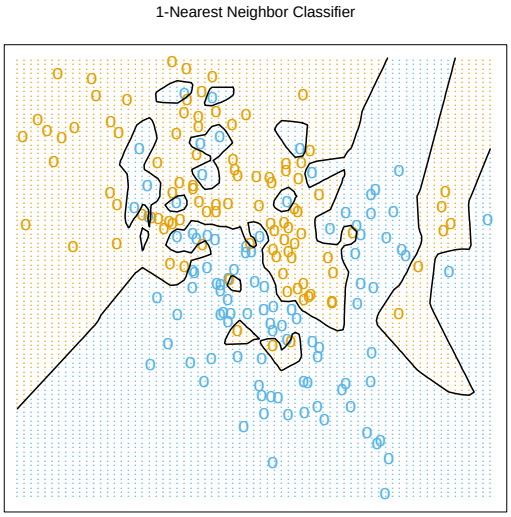

FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

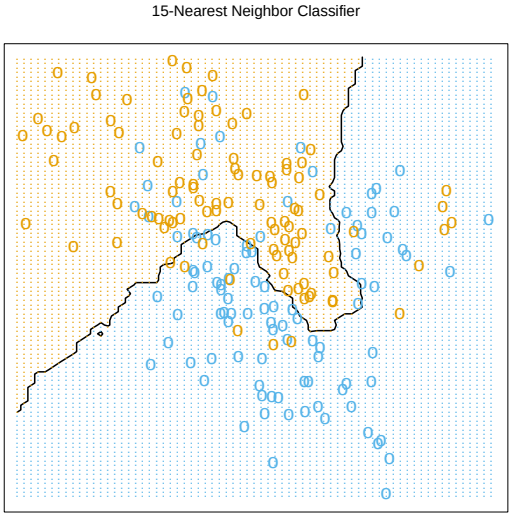

FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Bias-variance revisited

Note that if we pick a large value of k then we are smoothing/averaging over a large area. In the extreme case where

k is equal to the size of the training data our prediction will become a constant. Therefore – large k equals simple model, small k equals complex model.
Hence, if we pick k small we expect a small bias but large variance and if we pick k large we expect a large bias but small variance. We see in Figure "2.4" the usual "U" shaped curve. The left part of the figure corresponds to large k (small complexity) and the right part corresponds to small k (large complexity. On the left the test error is large due to a model that is too simple (too much averaging), whereas on the right it is large due to a large variance.

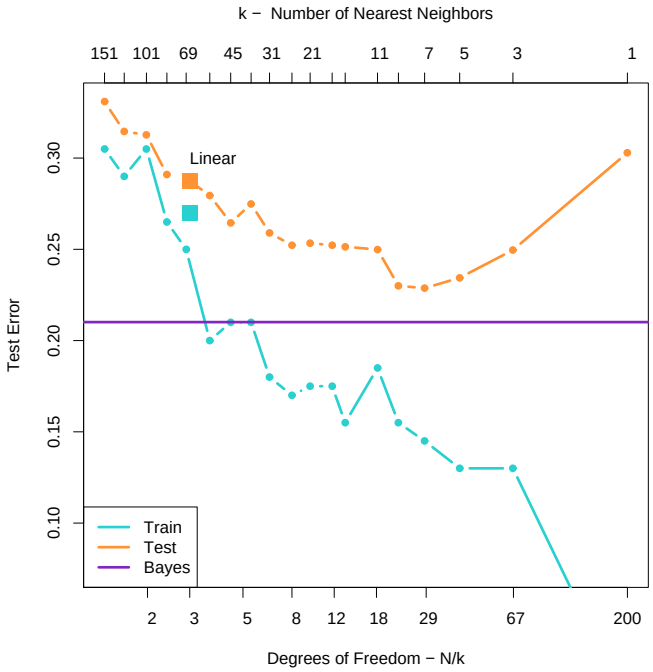

FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size* 200 *was used, and a test sample of size* $10,000$. *The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

Curse of dimensionality

According to Pedro Domingos: "Intuitions fail in high dimensions". This is also known as the curse of dimensionality (Bellman, 1961).

Claim 1: "Generalizing correctly becomes exponentially harder as the dimensionality grows because fixed-size training sets cover a dwindling fraction of the input space."

To see this, assume that our data is uniformly distributed in the box $[0, 1]^d$. Take the point at the center of this box, i.e., the point $(\frac{1}{2}, \cdots, \frac{1}{2})$, and draw a (small) box around it of side length r. What fraction of the total volume does this smaller box cover? It is r^d. Therefore, in expectation a fraction r^d of the data lies in this small box. Let the desired fraction be α. Then for $\alpha = 0.01$ in 10 dimensions we need r to be 0.63 and if we want to capture a fraction 0.1 of the data then we need $r = 0.8$ (recall the large box has a side length of only 1!). So we need to explore almost the whole range in each dimension!

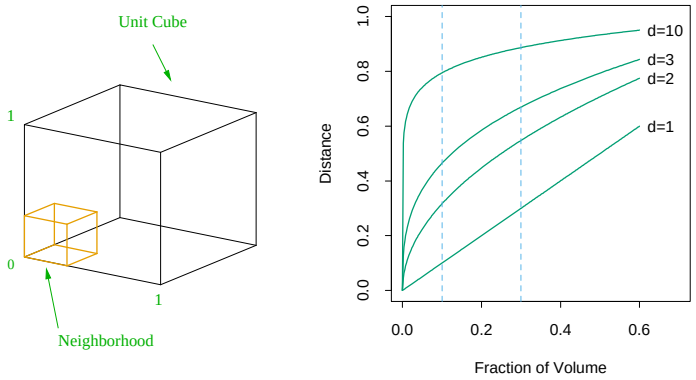

FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover* 80% *of the range of each coordinate to capture* 10% *of the data.*

We will see shortly that as a result we have to scale the number of samples exponentially in the dimension if we want our risk to stay constant.

Claim 2: In high-dimension, data-points are far from each other. Consequently, "as the dimensionality increases, the choice of nearest neighbor becomes effectively random."

Consider again N data points uniformly distributed in the box $[0,1]^d$. We consider a nearest-neighbor estimate at the point $(\frac{1}{2}, \cdots, \frac{1}{2})$. We claim that the median distance, call it r, from this point to the closest data point is,

$$\left(1 - \frac{1}{2}^{1/N}\right)^{1/d}$$

For $N = 500, d = 10$, this number is 0.52. So the box has to have a side length of half the one of the large box before we can hope to find a neighbor at all.

To see the above claim take a small box of side-length r centered at the point in the point $(\frac{1}{2}, \cdots, \frac{1}{2})$. What is the chance that a random sample inside the unit box is not inside this small box? This probability is $1-r^d$, since the small box covers a fraction r^d of the volume of the big box. Therefore, the chance that none of N iid samples fall inside the small box is equal to $(1-r^d)^N$. We want this probability to be $\frac{1}{2}$ (we are looking for the median). Solving for r confirms the above claim.

Analysis of nearest neighbor rule

Our aim is to analyze the simplest setting. We consider the nearest neighbor classifier and we will compare its perfor-

mance to the optimal (Bayes) classifier. This will confirm our intuition about the curse of dimensionality and tell us how good this classifier performs compared to the best we can hope for. In particular, if we have plenty of data we will see a simple and pleasing connection.

Here is our set-up. We assume that $\mathbf{x}$ takes values in $\mathcal{X} = [0,1]^d$ and that $\mathcal{Y} = \{0,1\}$ (binary classification). We assume further that there is some unknown distribution $\mathcal{D}$ on $\mathcal{Y} \times \mathcal{X}$ from which samples are drawn. Our training set S_t will consist of N iid samples. Associated to the distribution $\mathcal{D}$ there is the conditional probability

$$\mathbb{P}\{y = 1 \mid \mathbf{x}\}.$$

To ease our notation, let us introduce

$$\eta(\mathbf{x}) = \mathbb{P}\{y = 1 \mid \mathbf{x}\}.$$

In words, $\eta(\mathbf{x})$ is the conditional probability that the label is 1 given that the input is $\mathbf{x}$.

If we knew the distribution $\mathcal{D}$ (and hence $\eta(\mathbf{x})$) we would employ the classifier $f_\star$ which outputs

$$f_\star(\mathbf{x}) = \mathbf{1}_{\{\eta(\mathbf{x}\} > \frac{1}{2}\}}.$$

This is the *Bayes* classifier (also called maximum a posteriori or MAP classifier). It has the smallest probability of misclassification of any classifier, namely

$$L(f_\star) = \mathbb{E}_{\mathbf{X}\sim\mathcal{D}}[\min\{\eta(\mathbf{x}), 1 - \eta(\mathbf{x})\}].$$

It is instructive to compare the classification error of our nearest neighbor classifier to this optimal classifier.

It is clear that we cannot hope to perform well if there is no correlation between the "position" $\mathbf{x}$ and the associated label. E.g., consider the case where the label y is chosen randomly and independently from $\mathbf{x}$. It is then of no help to know the labels of points close by. So we need to make a suitable assumption regarding $\mathcal{D}$. Here is one way of doing this. We demand that the distribution $\mathcal{D}$ is such that for some constant c we have

$$|\eta(\mathbf{x}) - \eta(\mathbf{x}')| \leq c||\mathbf{x} - \mathbf{x}'||,$$

where on the right we have the Euclidean distance. In words, we ask that the conditional probability $\mathbb{P}\{y = 1 \mid \mathbf{x}\}$ (seen as a function of $\mathbf{x}$) is Lipschitz continuous with Lipschitz constant c. This seems a reasonable assumption: if we move a point only slightly, we want that the probability of this point to have an associated label $y = 1$ to vary only slightly as well.

Let $\mathcal{L}(f_\star)$ denote the risk (classification error) for this setting assuming that we use the Bayes classifier and let $\mathcal{L}(f_{S_t})$ denote the risk (classification error) for this setting if we use the nearest neighbor classifier.

Lemma.

$$\begin{aligned}\mathbb{E}_{S_t}[L(f_{S_t})] &\leq 2L(f_\star) + c\,\mathbb{E}_{S_t;\mathbf{x}\sim\mathcal{D}}[\|\mathbf{x} - nbh_{S_t,1}(\mathbf{x})\|] \\ &\leq 2L(f_\star)] + 4c\sqrt{d}N^{-\frac{1}{d+1}}.\end{aligned}$$

Before we see where this bound comes from, let us interpret this result. We see that the risk of the nearest neighbor classifier is upper bounded by twice the risk of the optimal

classifier plus a "geometric" term. This term is the average distance of a randomly chosen point to the nearest point in the given training set times the Lipschitz constant c.
It is very natural to have this second term present. Our assumption was that nearby points are likely to have the same label. So it is reasonable to expect in the bound a term that depends on the average distance.
So let us discuss what happens in certain limiting cases. Assume at first that we have a fixed dimension and that the size of the training data tends to infinity. Then the second term on the right will converge to zero and we see that in this case we have a risk which is at most twice the Bayes risk. So if we have plenty of data then we are doing well!
On the other hand, if we fix the size of the data, namely N, but increase the dimension we see from the bound that very quickly we should expect a large error. The second term on the right is proportional to $1/N$ to the power $1/(d+1)$ (there is an extra term $4c\sqrt{d}$ in the bound but this is relatively minor and hence let us ignore it). More precisley, in order to keep the second term on the right constant, lets say α, $\alpha << 1$, we need to let N grow like $(1/\alpha)^{d+1}$, i.e., exponential in the dimension. This is another way of seeing the curse of dimensionality.

Proof. Let us now explain how to derive such a bound. We follow the proof in Chapter 19 of the "Understanding Machine Learning" book. You can find a more detailed explanation there.
We start with the second inequality. How do we bound $\mathbb{E}_{S_t;\mathbf{x}\sim\mathcal{D}}[\|\mathbf{x} - \mathrm{nbh}_{S_t,1}(\mathbf{x})\|]$ by $4\sqrt{d}N^{-\frac{1}{d+1}}$? Take the unit

cube $[0,1]^d$. Recall that this is the space of inputs. Cut this "large" cube into small little cubes of side length ϵ. Consider now what happens when we want to predict a label for a "fresh" input.

Consider the cube which contains $\mathbf{x}$. If we are lucky, the same cube also contains elements from the training set S_t. In this case $\mathbf{x}$ has a neighbor in S_t at distance at most $\sqrt{d}\epsilon$. This is good since by our assumption a short distance leads to good predictions. But if $\mathbf{x}$ lies in a cube which contains no such element from the training data, then its nearest neighbor might be much further, at worst at a distance $\sqrt{d}$. In this case the prediction is probably not very reliable.

So what is the chance that a randomly chosen point $\mathbf{x}$ ends up in a specific box and this box does not have a training point in there?

If the probability of $\mathbf{x}$ landing in a particular box i is lets say $\mathbb{P}_i$, then the chance that none of the N training symbols are in box i is $(1-\mathbb{P}_i)^N$. This is the main step in this proof. We do not know the distribution $\mathcal{D}$ and hence cannot determine the probabilities $\mathbb{P}_i$. But it turns out that this is not important. No matter how the probability is distributed over the boxes we are fine. The reason is simple. If $\mathbb{P}_i$ is large (i.e., this case happens often) we are fine since then it is very likely that we also have at least one training point in this box. But if $\mathbb{P}_i$ is small (and hence also the probability that we have a training point in there is small) we are fine since by definition this does not happen very often.

The rest is calculus, carefully choosing the right scaling for ϵ in order to get a good bound.

It still remains to explain the term $2L(f_\star)$. Consider the following experiment. We are given two points $\mathbf{x}$ and $\mathbf{x}'$, both elements of $[0,1]^d$. Assign to these two points labels y and y' according to the distribution $\eta(\cdot)$. What is the chance that these two labels are the same? We claim that

$$\mathbb{P}\{y \neq y'\} \leq 2\eta(\mathbf{x})(1-\eta(\mathbf{x})) + c\|\mathbf{x}-\mathbf{x}'\|. \qquad (1)$$

This is easily explained. Assume at first that we draw the two labels y and y' independently but according to the *same* conditional probability $\eta(\mathbf{x})$. In this case the probability that the two labels are different is exactly equal to $2\eta(\mathbf{x})(1-\eta(\mathbf{x}))$. But according to our model we draw the label for $\mathbf{x}$ according to the conditional probability $\eta(\mathbf{x})$ and the label for $\mathbf{x}'$ independently but according to the conditional probability $\eta(\mathbf{x}')$. These two quantities are in general not the same. But we have

$$\begin{aligned}
&\eta(\mathbf{x})(1-\eta(\mathbf{x}')) + \eta(\mathbf{x}')(1-\eta(\mathbf{x})) \\
=&2\eta(\mathbf{x})(1-\eta(\mathbf{x})) + (2\eta(\mathbf{x})-1)(\eta(\mathbf{x})-\eta(\mathbf{x}')) \\
\leq&2\eta(\mathbf{x})(1-\eta(\mathbf{x})) + (\eta(\mathbf{x})-\eta(\mathbf{x}')) \\
\leq&2\eta(\mathbf{x})(1-\eta(\mathbf{x})) + c\|\mathbf{x}-\mathbf{x}'\|.
\end{aligned}$$

Consider now the following experiment. Draw a set S_t according to the distribution $\mathcal{D}$ but hide the labels y_n and only reveal the inputs $\mathbf{x}_n$. Then draw one more "fresh" sample $(y,\mathbf{x})$ but hide also in this case the label y and only reveal $\mathbf{x}$. Find the point in S_t that is closest to $\mathbf{x}$, call it $\mathrm{nbh}_{S_t,k}(\mathbf{x})$. Now reveal the label y associated to $\mathbf{x}$ as well as the label $y_{i\mathrm{nbh}_{S_t,k}(\mathbf{x})}$ associated to this closest point $\mathrm{nbh}_{S_t,k}(\mathbf{x})$. What is the probability that these two labels do not agree?

This is exactly the risk $\mathbb{E}_{S_t}[L(f_{S_t})]$. And according to (1) we can bound this probability by averaging the right hand side over all choices of $\mathbf{x}$ and $\text{nbh}_{S_t,k}(\mathbf{x})$. The average of the term $2\eta(\mathbf{x})(1-\eta(\mathbf{x}))$ is upper bounded by $2\mathcal{L}(f_\star)$ since $2\eta(\mathbf{x})(1-\eta(\mathbf{x})) \leq 2\min\{\eta(\mathbf{x}), 1-\eta(\mathbf{x})\}$. And the average of the term $c\|\mathbf{x}-\mathbf{x}'\|$ is $c\,\mathbb{E}_{S_t;\mathbf{x}\sim\mathcal{D}}[\|\mathbf{x}-\text{nbh}_{S_t,1}(\mathbf{x})\|]$.

□

Discussion

(Taken from HTF). We will see (in the next few lectures) that there is a whole spectrum of models between the rigid linear models and the extremely flexible 1-NN model. Each model comes with their own assumptions and biases.
(Based on Domingos). You might think that gathering more input variables never hurts, since at the worst they provide no new information about the output. But in fact their benefits may be outweighed by the curse of dimensionality.