2DI70 - Statistical Learning Theory

Nearest neighbor classification and
handwritten digit classification

1 Introduction

Sometimes, simple ideas can be surprisingly good. This is the case with one of the oldest, but still rather popular *learning rule*, known as the *k-nearest neighbor* rule (abbreviated k-NN in this document). Consider the setting of supervised learning. Suppose you have a training data set $\{X_i, Y_i\}_{i=1}^n$, where $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$, where $\mathcal{X}$ should be a metric space (that is, a space endowed with a way to measure distances). As usual, our goal is to learn a prediction rule $f : \mathcal{X} \to \mathcal{Y}$ that is able to do "good" predictions on unseen data.

The idea of k-NN is remarkably simple. Given a point $x \in \mathcal{X}$ for which we want a prediction, we simply look for the k "closest" points in the training set and make a prediction based on a majority vote (classification) or average (regression) of the neighbor labels. That is as simple as that. Computationally this might seem cumbersome, particularly for large datasets. But one can use clever computational tricks to ensure this can be done quickly.

In this assignment, which is divided in two parts, you will: (i) get a first-hand experience with this method by implementing it and choosing the "right" set of tunable parameters in a sound way; (ii) analyze the performance of this method in some generality and get a better understanding why it is sensible.

To make the explanation more concrete let us consider the problem of handwritten digit classification (which is the topic of part I): given a low resolution image of a handwritten digit we would like to classify it as one of the digits in $\{0, 1, \ldots, 9\}$. More specifically our images have 28x28 pixels, each pixel taking values in $\{0, 1, 2, \ldots, 255\}$. Therefore $\mathcal{X} = \{0, 1, \ldots, 255\}^{28\times 28}$ and $\mathcal{Y} = \{0, 1, \ldots, 9\}$.

2 The k-NN rule

Let $d : \mathcal{X} \times \mathcal{X} \to [0, +\infty)$ be a metric[1] in $\mathcal{X}$. Let $x \in \mathcal{X}$ be an arbitrary point in $\mathcal{X}$. Given x let us reorder each pair of the training data as

$$\left(X_{(1)}(x), Y_{(1)}(x)\right), \ \left(X_{(2)}(x), Y_{(2)}(x)\right), \ldots, \left(X_{(n)}(x), Y_{(n)}(x)\right) \ ,$$

so that

$$d(x, X_{(1)}(x)) \leq d(x, X_{(2)}(x)) \leq \cdots \leq d(x, X_{(n)}(x)) \ .$$

[1] A metric or distance is a function that must satisfy the following properties: (i) $\forall x \in \mathcal{X}\ d(x, x) = 0$; (ii) $\forall x, y \in \mathcal{X}\ d(x, y) = d(y, x)$ (symmetry); (iii) $\forall x, y, z \in \mathcal{X}\ d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

Note that the ordering depends on the specific point x (hence the cumbersome notation), and might not be unique. In that case we can break ties in some pre-defined way (e.g., if to points are at equal distance from x the point that appears first in the original dataset will also appear first in the ordered set). The k-NN rule (for classification) is defined as

$$\hat{f}_n(x) = \arg\max_{y \in \mathcal{Y}} \left\{ \sum_{i=1}^{k} \mathbf{1} \left\{ Y_{(i)}(x) = y \right\} \right\} .$$

In other words, just look among the k-nearest neighbors and choose the class that is represented more often. Obviously, there might be situations where two (or more) classes appear an equal number of times. In such situations one can break these ties according to a pre-specified rule (e.g., take the most common class that appears first in $\mathcal{Y}$).

The performance of the method described above hinges crucially on the choice of two parameters: k, the number of neighbors used for prediction and; $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the distance metric used to define proximity of two points in the feature space. There are many possible choices for d, and a naïve but sensible starting point is to consider the usual Euclidean distance: if $x, y \in \mathbb{R}^d$ then the Euclidean distance is simply given by $\sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$.

3 The MNIST dataset

This MNIST dataset[2] is a classical dataset to demonstrate machine learning methods, and is still used often as a benchmark to compare methodologies. This dataset is provided as *comma-separated value* (csv) files in CANVAS. The training set `MNIST_train.csv` consists of 60000 images of handwritten digits and the corresponding label (provided by a human expert). The test set `MNIST_test.csv` consists of 10000 images of handwritten digits and the corresponding labels (this file will be provided only closer to the assignment deadline). In addition, in CANVAS you will also find two smaller training and test sets, `MNIST_train_small.csv` (3000 examples) and `MNIST_test_small.csv` (1000 examples). These will be used for a large part of the assignment, to avoid the struggles associated with large datasets and to test out your implementations.

The format of the data is as follows: each row in the `.csv` file has 785 entries and corresponds to a single example. The first entry in the row is the "true" label, in $\mathcal{Y} = \{0, 1, \ldots, 9\}$ and the 784 subsequent entries encode the image of the digit – each entry corresponding to a pixel intensity, read in a lexicographical order (left-to-right then top-to-bottom). Pixel intensities take values in $\{0, 1, \ldots, 255\}$. The Matlab function `showdigit.m` in CANVAS will take as input a row of this data and display the corresponding digit image. Figure 3 shows several examples from `MNIST_train.csv`.

Ultimately the goal is to minimize the probability of making errors. For the purposes of this assignment we will use simply the 0/1 loss. This means that the empirical risk is simply the average number of errors we make. If $\{X'_i, Y'_i\}_{i=1}^{m}$ denotes the pairs of features/labels

[2] See `http://yann.lecun.com/exdb/mnist/` for more details and information.

Figure 1: First five examples from `MNIST_train.csv` and the corresponding labels provided by a human expert.

in a test set and $\{\hat{Y}_i'\}_{i=1}^m$ denotes the corresponding inferred labels by the k-NN rule then the empirical risk on the test set is given by $\frac{1}{m}\sum_{i=1}^m \mathbf{1}\left\{\hat{Y}_i' \neq Y_i'\right\}$.

PART I - Computational Assignment

The goal of the first part of the assignment is to implement "from scratch" a nearest neighbor classifier. This means you will not be allowed to use existing libraries and implementations of nearest neighbors, and should make only use of standard data structures and mathematical operations[3]. The only "exception" is that you are allowed to use a sorting subroutine (i.e., a function that, given a vector of numerical values, can sort them in ascending order and give the correspondent reordered indexes). The rational for the above restrictions is for you to experience what are the critical aspects of your implementation, and understand if it is scalable to big datasets. For this assignment you are allowed to use any language or command interpreter (preferably a high-level language, but not necessarily so). You will not be judged on your code, but rather on your choices and corresponding justification.

You should prepare a report (in English) and upload it via CANVAS. The report should be self-contained and document in a clear and concise way the problem, your solution and results. It is import that for you to have a critical attitude, and comment on the your choices and results. The report for part I should be submitted as a single .pdf file. In addition, submit a separate file with the code/script you used (you will not be graded on this, but if needed we might look at it to better understand the results in your report). In your report you should do the following experiments and answer the questions below.

a) Write down you implementation of k-NN neighbors (using as training data `MNIST_train_small.csv`) and report on its accuracy to predict the labels in both the training and test sets (respectively `MNIST_train_small.csv` and `MNIST_test_small.csv`). For this question use the simple Euclidean distance. Make

[3]This means that, libraries encoding useful data-structures are allowed, as long as these are not specifically targeting nearest neighbors.

a table of results for $k \in \{1, \ldots, 20\}$, plot your the empirical training and test loss as a function of k, and comment on your results.

b) Obviously the choice of the number of neighbors k is crucial to obtain good performance. This choice must be made WITHOUT LOOKING at the test dataset. Although one can use rules-of-thumb, a possibility is to use cross-validation. *Leave-One-Out Cross-Validation* (LOOCV) is extremely simple in our context. Implement LOOCV to estimate the risk of the k-NN rule for $k \in \{1, \ldots, 20\}$. Report these LOOCV risk estimates[4] on a table and plot them as well the empirical risk on the test dataset (that you obtained in (a)). Given your results, what would be a good choice for k?

c) Obviously, the choice of distance metric also plays an important role. Consider a generalization of the Euclidean distance, namely L_p distances (also known as Minkowski distances). For $x, y \in \mathbb{R}^d$ define

$$d_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p},$$

where $p \geq 1$. Use leave-one-out cross validation to simultaneously choose a good value for k and $p \in [1, 15]$.

d) **(this question is more open)** Building up on your work for the previous questions suggest a modification of the distance metric or some pre-processing of the data that you consider appropriate to improve the performance of the NN method. Note that, any choices you make should be done solely based on the training data (that is, do not optimize the performance of your method on the test data). Clearly justify ALL the choices you made and describe the exact steps you took. Someone reading your report should be able to exactly replicate your results.

Now that you implemented and tested your methodologies in a smaller scale, let us see how these methods scale to the real problem. For the remaining questions you will use the full MNIST training and test sets.

e) Make use of either the Euclidean distance or d_p with your choice of p in part (c) (use only one or the other). Determine a good value for k using leave-one-out cross validation when considering the full training set (60000 examples). Was your implementation able to cope with this large amount to data? Did you have to modify it in any way? If so, explain what you did. What is the risk estimate you obtain via cross-validation?

[4] Recall that these estimates use only the information on the training dataset.

f) **(it is only possible to answer this question after I provide you the file `MNIST_test.csv`)** Using the choice of k in part (e) compute the loss of your method on the test set provided. How does this compare with the cross-validation estimate you computed in (e)? Would you choose a different value for k had you been allowed to look at the test dataset earlier?

g) **Bonus question:** each training example is currently a high-dimensional vector. A very successful idea in machine learning is that of dimensionality reduction. This is typically done in an unsupervised way - feature vectors are transformed so that most information is preserved, while significantly lowering their dimension. A possibility in our setting is to use Principal Component Analysis (PCA) to map each digit image to a lower dimensional vector. There is an enormous computational advantage (as computing distances will be easier) but there might be also an advantage in terms of statistical generalization. Use this idea for this problem and choose a good number of principal components to keep in order to have good accuracy (again, this choice should be solely based on the training data). Document clearly all the steps of your procedure. In this question you are allowed to use an existing implementation of PCA or related methods.

PART II - Analysis of k-NN classification

In this part of the assignment you will analyze the k-NN rule and get a better understanding why and when it works well. This part of the assignment is a guided proof, and you will need to fill in the details in the argument below. The specific questions asked are presented inside boxes, so it is easier for you to see what we expect you to do. In any case, in your report you should essentially write down the statement you show in the end, and the various steps in the proof, and in concise but complete way. Submit your report in a single .pdf file via CANVAS.

For simplicity we focus on the case of binary classification, so there are only two classes. It is possible to extend the analysis to multiple classes (and also to regression) but we are not going to do this here. In what follows $\mathcal{Y} = \{0, 1\}$ and we are considering the 0/1 loss, so we are precisely under the setting of Section 2.1 of the lecture notes. We assume the training data $D_n = \{X_i, Y_i\}_{i=1}^n$ is an independent and identically distributed sample from an unknown distribution $\mathbb{P}_{XY}$. The risk of the k-NN rule is given by

$$R(f) = \mathbb{P}\left(\hat{f}_n(X) \neq Y \middle| \hat{f}_n\right) ,$$

where $(X, Y) \sim \mathbb{P}_{XY}$ and (X, Y) is independent of D_n. In this setting the Bayes risk R^* is attained by the Bayes classifier $f^*(x) = \mathbf{1}\{\eta(x) > 1/2\}$, where $\eta(x) = \mathbb{P}(Y = 1|X = x)$.

To begin our analysis note first that k-NN is actually a *plug-in* rule, that first estimates $\eta(x)$ by $\hat{\eta}(x)$ and then *plugs-in* this estimate in the expression for the Bayes classifier, to obtain

$$\hat{f}_n(x) = \mathbf{1}\{\hat{\eta}(x) > 1/2\} .$$

More specifically, and using the notation in Section 2 we have

$$\hat{\eta}(x) = \frac{1}{k}\sum_{i=1}^{k} Y_{(i)}(x) . \tag{1}$$

a) Check that, for every $x \in \mathcal{X}$, we have

$$\mathbf{1}\{\hat{\eta}(x) > 1/2\} = \hat{f}_n(x) = \arg\max_{y\in\{0,1\}} \left\{ \sum_{i=1}^{k} \mathbf{1}\left\{Y_{(i)}(x) = y\right\} \right\} .$$

Given that nearest neighbors classification is a consequence of a regression rule (which is written in a rather simple way) it makes sense to relate the classification performance of k-NN to the performance of the corresponding regression rule.

b) We will focus on the expected excess risk. Making use of Lemma 2.1.1. of the lecture notes, show that

$$\mathbb{E}[R(\hat{f}_n)] - R^* \leq 2\mathbb{E}\left[\int_{\mathcal{X}} |\hat{\eta}(x) - \eta(x)| d\mathbb{P}_X(x)\right] = 2\mathbb{E}\left[|\hat{\eta}(X) - \eta(X)|\right] \ .$$

In conclusion, if we do a good job in regression (with absolute loss) we will also do a good job in classification.

As with most methods we have studied in class, there is a natural trade-off between approximation and estimation. In the case of k-NN the larger k is, the more *averaging* and therefore the better we cope with uncertainty in data. On the other hand, for larger k size of the neighborhoods around any point will also grow. If they are too large then the approximation of $\eta(x)$ will be poor, as we will be averaging the labels of examples that are not very much alike. So it makes sense to decompose the regression error in two components, one corresponding to approximation error and one corresponding to estimation error. For the absolute loss we can do in a very natural way, by using the triangle inequality:

$$\mathbb{E}\left[|\hat{\eta}(X) - \eta(X)|\right] \leq \underbrace{\mathbb{E}\left[|\hat{\eta}(X) - \bar{\eta}(X)|\right]}_{\text{estimation error}} + \underbrace{\mathbb{E}\left[|\bar{\eta}(X) - \eta(X)|\right]}_{\text{approximation error}} ,$$

where we need to choose $\bar{\eta}$ appropriately.

There are two sources of randomness in the rule we are considering - that coming from the feature vectors $\{X_i\}_{i=1}^n$ and that coming from the corresponding labels $\{Y_i\}_{i=1}^n$. We will begin our analysis by *conditioning* on the feature vectors and understanding the role of the randomness on the labels. Once that is clearer out, we can deal with the randomness of the features. With this in mind it makes sense to define

$$\begin{aligned}
\bar{\eta}(x) &= \mathbb{E}\left[\hat{\eta}(x) \,|\{X_i\}_{i=1}^n\right] \\
&= \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} Y_{(i)}(x) \,|\{X_i\}_{i=1}^n\right] \\
&= \frac{1}{k}\sum_{i=1}^{k} \mathbb{P}\left(Y_{(i)}(x) = 1 \,|\{X_i\}_{i=1}^n\right) \\
&= \frac{1}{k}\sum_{i=1}^{k} \eta(X_{(i)}(x)) \ .
\end{aligned}$$

c) We begin by studying the estimation error. Let $x \in \mathcal{X}$. Show first that

$$\mathbb{E}\left[(\hat{\eta}(x) - \bar{\eta}(x))^2 \,|\{X_i\}_{i=1}^n\right] \leq \frac{1}{4k} \ .$$

Use that result, together with the Cauchy-Schwarz inequality to show that

$$\mathbb{E}\left[|\hat{\eta}(x) - \bar{\eta}(x)|\right] \leq \frac{1}{2\sqrt{k}} \ .$$

Now that the estimation error is taken care of, we can focus on the approximation error. If we want finite-sample bounds (so bounds that are not only asymptotic) we need to make extra assumptions on $\mathbb{P}_{XY}$.

For concreteness, we will consider the case where the features are real numbers. The analysis can easily be extended beyond that, but that is out to the scope of this assignment. Let $M > 0$ and assume $\mathcal{X} = [-M, M]$. Assume X is a continuous random variables with probability density function f_X. Finally, let us consider a Lipschitz assumption on η, namely:

Assumption 1. *There exists a constant $C > 0$ such that, for all $x, y \in \mathcal{X}$*

$$|\eta(x) - \eta(y)| \leq C|x - y| \ .$$

This means that points that are close to each other have similar values for η.

d) Show that for a given $x \in \mathcal{X}$ we have

$$|\bar{\eta}(x) - \eta(x)| \leq \frac{C}{k} \sum_{i=1}^{k} |X_{(i)}(x) - x| \ .$$

Note this is a random quantity.

e) From this expression we see that the approximation error will be large if at least one of the neighbors is far from x. Define $p(x, r) = \mathbb{P}(|X - x| \leq r)$ and let $r > 0$ be arbitrary. Show that, provided $k \leq np(x, r)$ we have

$$\mathbb{P}\left(\exists i \in \{1, \ldots, k\} \text{ such that } |X_{(i)}(x) - x| > r\right) \leq \exp\left(-\frac{(np(x,r) - k)^2}{2np(x,r)}\right) \ .$$

Hint: to answer this question it might be convenient to define $N(x, r) = |\{i : X_i \in [x - r, x + r]\}|$. In words, $N(x, r)$ is number of training points inside the interval $[x - r, x + r]$. Relate the above probability to $N(x, r)$ and use the following relative bounds for binomial random variables.

Lemma 1. [a] *Let $B \sim Bin(n,p)$ be a binomial random variable and $0 \leq \varepsilon \leq 1$. Then*

$$\mathbb{P}(B \geq (1+\varepsilon)\mathbb{E}[B]) \leq e^{-\varepsilon^2 \mathbb{E}[B]/3} \ ,$$

and

$$\mathbb{P}(B \leq (1-\varepsilon)\mathbb{E}[B]) \leq e^{-\varepsilon^2 \mathbb{E}[B]/2} \ .$$

[a] See for instance Hagerup, T. and Rüb, C., "A Guided Tour of Chernoff Bounds", Information Processing Letters 33 (1989/90) pp. 305–308.

With the above result at hand we might be tempted to choose k to be just a little smaller than $np(x,r)$ for some value of r, as we would like to choose k as large as possible but still ensuring the k-neighborhoods of points are small. The problem is that k must be the same for all points $x \in \mathcal{X}$. To deal with this issue let us first make an extra assumption

Assumption 2. *The density of X, denoted by $f_X : \mathcal{X} \to [0,\infty)$, is such that $f_X(x) \geq f_{\min}$ for some $f_{\min} > 0$. Therefore*

$$p(x,r) = \mathbb{P}(|X - x| \leq r) = \mathbb{P}(X \in [x-r, x+r]) \geq 2r f_{\min} \ .$$

This suggests the following choice for k:

$$k = nr f_{\min} \ ,$$

where we still need to decide on what is the best choice for r.

f) Based on the above choice of k show that the approximation error is bounded by

$$\mathbb{E}\left[|\bar{\eta}(x) - \eta(x)|\right] \leq Cr + e^{-\frac{nr f_{\min}}{4}} \ .$$

g) Making use of all the results derived so far, what is a good choice for r? Use this choice of r to determine how to choose k and give an upper bound on the expected excess risk of $\hat{f}_n$. That is, give an upper bound on

$$\mathbb{E}[R(\hat{f}_n)] - R^* \ .$$

At what rate, as a function of the sample size n, does the expected excess risk decay? At what rate should we increase k?

h) At this point it is useful to reflect upon our results and contrast them with the computational experience of Part I. Compare the "best" choice for k you came up for classification in the MNIST setting with the choice suggested by the theory (this should be a qualitative comparison only). In light of your findings, do you think Assumption 1 is reasonable? Is our analysis to loose or pessimistic?

Bonus Questions

The questions below are not mandatory, and doing them could get you extra credit in the assignment (besides some satisfaction).

i) **Bonus:** To answer the questions (f) and (g) we made the extra Assumption 2, that the density of X is lower-bounded by $f_{\min} > 0$. However, this is rather strong, and we would like to drop this assumption. Modify the arguments to drop this assumption and get a bound on

$$\mathbb{E}[R(\hat{f}_n)] - R^* \ .$$

Is this qualitatively the same bound you obtained in the previous question? If not, do you think your analysis can be improved to get the same error decay rate as before, but without Assumption 2? Explain your reasoning.

j) **Bonus:** The rates of the error decay obtained with the above analysis are essentially optimal, meaning that, under the same assumptions, not other method can substantially outperform k-NN. Although there is nothing fundamentally wrong with the setting considered above, it does not match closely to what one expects in practice. Actually, all we care about in classification is to determine, for every $x \in \mathcal{X}$, on which side of $1/2$ is the value of $\eta(x)$. Therefore, if $\eta(x)$ is far from $1/2$ we can have a lousy estimate $\hat{\eta}(x)$, as long as it is on the correct side of $1/2$.

With this in mind take $\mathcal{X} = [-M, M]$ and make the following two assumptions:

Assumption 3. *The is a constant $C > 0$ so that, for all $x \in \mathcal{X}$ we have*

$$|\eta(x) - 1/2| \geq C \ .$$

Assumption 4. *Assume the Bayes decision boundary is not terribly complicated. The Bayes decision boundary is given by*

$$\partial G^* = \{x \in \mathcal{X} : \forall \varepsilon > 0 \ \exists y_1, y_2 \in [x - \varepsilon, x + \varepsilon] \ : \eta(y_1) < 1/2 \text{ and } \eta(y_2) > 1/2\} \ .$$

Assume ∂G^* has only a finite number of points (so the function η "jumps" across $1/2$ a finite number of times.

Using the above two assumptions characterize the expected excess risk of $\hat{f}_n$. Note that, to get a good characterization you will not be able to use the result you derived in (b), and will have to work directly with Lemma 2.1.1.