DATA11002 Introduction to Machine Learning, Fall 2017
Exercise set 6

Due November 13th–15th.

Read pp. 316–321 (bootstrap, bagging, and random forests) in the course book.

Problem 1 (3+3 points) k-nearest neighbor classifier

Even though the library `class` in R provides a ready-made implementation of the k-NN classifier, you get to do it yourself in this exercise (Yay!). If you didn't do it last week, download the MNIST handwritten digit database from `http://yann.lecun.com/exdb/mnist/`, and load the data into R.[1]

(a) (3 points) Use the first 5 000 training instances and the first 1 000 test instances only, and discard the rest. (Unless you have a supercomputer or very much patience.) Compute all pairwise Euclidean distances, $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{784}(x_{ik} - x_{jk})^2}$, where i runs through the 5 000 training instances, and j runs through the 1 000 test instances. Verify that the distance between the first training instance and the first test instance equals about 2395.8.

Hint: Function `dist` in library `proxy`[2] does this very nicely but you can also write `for` loops.

(b) (3 points) Having stored the pairwise distances into a 5 000 $\times$ 1 000 distance matrix so that you don't have to recalculate them again later, classify each test instance by finding the k training instances nearest to it, and choosing the majority class among them.

Compute and plot the test set accuracy of the k-NN classifier with $k = 1, \ldots, 50$.

Hint: Here's a way to get the most common entry in a list: `names(sort(table(...), decreasing=TRUE))[1]`, where you should write the name of the list at `...`.

[1] Here's the script again to access the data in R: `https://gist.github.com/brendano/39760`. Just remember to put the files in folder `mnist` and unzip them. Also check the further instructions in last week's exercises to make sure that loading the data was successful.

[2] You can install libraries using `install.packages("proxy")`, etc.

Problem 2 (2+3+2+5 points) Principal component analysis (PCA)

Since we're working with the MNIST data, why don't we continue with it! In this exercise, we'll use all the 60000 training instances and store them in an $n \times p$ matrix (where the sample size $n = 60000$, and the dimension $p = 784$). However, you may actually want to practice the following steps with a small subset first.

(a) (2 points) Center the data by subtracting the overall mean vector from each row. Normalize the data by dividing each column by its standard deviation unless it is zero. Denote the resulting matrix by X. Now construct the $p \times p$ empirical covariance matrix $X^T X/n$. Check that it's diagonal entries are all equal to either 0 or 1 (since some of the columns are all zeros).

Hint: R may be incredibly unwieldy. Subtraction of the column means is one such situation. To save you some trouble: `t(t(X) - rep(colMeans(X), n))` does this for you. A similar expression can be used to divide each row by its standard deviation.

(b) (3 points) Perform an eigendecomposition of the covariance matrix. Study the first two eigenvectors (i.e., those with the largest eigenvalues). They are 784 dimensional vectors. Try plotting them as 28×28 images. Give an interpretation to the vectors.

Hint: Recall the `eigen` function in R. For the interpretation, see Sec. 10.2.1 of the textbook, and recall that in PCA, we represent the data as a linear combination of the principal component vectors.

(c) (2 points) Project the data onto the first two principal components, and plot the data. Use colors or symbols to distinguish images representing different digits (0, ..., 9).

(d) (5 points) Reduce the dimensionality of the data to $q = 5$ using the first 5 principal components. Such low-dimensional data makes the k-NN classifier from Exercise 1 much faster (although it is still $\mathcal{O}(n_{\text{train}} n_{\text{test}} q)$, where n is the bigger problem).

Evaluate the accuracy of the k-NN classifier with $k = 5$ using all the 60000 instances as training data and the first 1000 test instances as test data. It is important to notice that the test data should be centered and normalized by subtracting the mean of the *training data* and dividing the columns by the respective standard deviations of the *training data*. After this, project the data onto the same $q = 5$ principal component vectors.

Repeat the procedure with $q = 10, 20, 40, 80$, and evaluate the test set accuracy. If everything is done correctly, you should reach accuracy above 95 %.

Problem 3 (2+2+2 points) Resampling

(a) (2 points) Consider a sample of size n drawn from a univariate normal distribution, $\mathcal{N}(\mu, \sigma^2)$. The sample average $\bar{x} = (1/n)\sum_{i=1}^{n} x_i$ is a good estimator of the mean μ. A closed form confidence interval exists in case the variance σ^2 is known:

$$\left[\bar{x} - 1.96\frac{\sigma}{\sqrt{n}}, \bar{x} + 1.96\frac{\sigma}{\sqrt{n}}\right]$$

which contains the true mean with at least 95 % probability.

Empirically validate this confidence interval by repeating the trial $m = 1000$ times. Use the sample size $n = 100$ and parameters $\mu = 0, \sigma^2 = 100$. For each trial, check whether the above confidence interval contains the true mean μ. This should be the case in about 95 % of the samples.

(b) (2 points) Construct a bootstrap confidence interval for the mean using one of the samples from item *a* by resampling $K = 1000$ bootstrap samples D_j^* and taking the 2.5th and 97.5th percentiles.

Repeat $m = 1000$ times to see how often the obtained bootstrap confidence interval contains the true mean μ. Again, this should be about 95 %.

Remark: Notice that here the bootstrap method is not really necessary since a closed form confidence interval is available. However, this is the case in special cases only where the underlying distribution and the estimator are "nice" enough. For more complex distributions and other estimators (of other parameters, such as the median), the bootstrap method may be the only available technique.

(c) (2 points) Construct a bootstrap confidence interval for the median and the maximum. (Does this make sense?)