Machine Learning 2017/2018
Home Assignment 3

Yevgeny Seldin, Christian Igel
Department of Computer Science, University of Copenhagen

The deadline for this assignment is **12 December 2017**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your source code in the PDF file.
- A .zip file with all your solution source code (Matlab / R / Python scripts / C / C++ / Java / etc.) with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in the speed grader. Zipped pdf submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Your code should include a README text file describing how to compile and run your program, as well as a list of all relevant libraries needed for compiling or using your code.

1 To Split or Not To Split? (And How to Split.)

1. You have a validation set $S_{\texttt{val}}$ of n samples and M hypotheses, $h_1, \ldots, h_M$ (for example, K-NN classifiers with $K \in \{1, \ldots, M\}$). You test all the

hypotheses on the validation set and pick the one with the smallest validation error $\hat{L}(h, S_{\texttt{val}})$, lets call it $\hat{h}^*$ (the hat indicates that it is the best according to the validation error, which does not necessarily mean that it has the smallest expected error). Provide a bound for the expected loss $L(\hat{h}^*)$, which holds with probability greater than $1 - \delta$.

2. A fellow student proposes to split the validation set $S_{\texttt{val}}$ into M parts of size n/M each and test each hypothesis h_i on the corresponding subset of the validation set, lets call it $S^i_{\texttt{val}}$. And then you pick $\hat{h}^*$ with the minimal validation error $\hat{L}(h_i, S^i_{\texttt{val}})$. Is it a good idea to do the split? Provide a bound for the expected loss $L(\hat{h}^*)$ based on the new procedure and argue from the perspective of the bound. (The bound should hold with probability at least $1 - \delta$.)

3. Another fellow student proposes a different procedure. You split the validation set into two sets, $S^1_{\texttt{val}}$ and $S^2_{\texttt{val}}$, each of size $n/2$. First you test all M hypotheses on $S^1_{\texttt{val}}$ and pick the best one, $\hat{h}^*$, according to $\hat{L}(h, S^1_{\texttt{val}})$. Then you calculate $\hat{L}(\hat{h}^*, S^2_{\texttt{val}})$ (i.e., you test it on the second half of the validation set) and you report a bound on $L(\hat{h}^*)$ based on $\hat{L}(\hat{h}^*, S^2_{\texttt{val}})$. Is it a good idea? Specifically:

   (a) What will be the bound on $L(\hat{h}^*)$ according to this procedure? (The bound should hold with probability at least $1 - \delta$.)

   (b) Imagine that you have followed the procedure in Point 1 and your fellow student have followed the procedure in Point 3 and for the sake of comparison imagine that $\hat{L}(\hat{h}^*, S_{\texttt{val}})$ in your case is equal to $\hat{L}(\hat{h}^*, S^2_{\texttt{val}})$ in the fellow student's case. Who will have a tighter bound on $L(\hat{h}^*)$? Or, more precisely, for what values of M (as a function of δ) your bound will be tighter and for what values of M it will be looser?

   (c) Can you spot any drawbacks of the procedure in Point 3? [Assuming that M is sufficiently large, so that the comparison in Point 3b is in favor of the procedure in Point 3.]

4. Now you are allowed to split $S_{\texttt{val}}$ into $S^1_{\texttt{val}}$ and $S^2_{\texttt{val}}$, so that the number of samples in $S^1_{\texttt{val}}$ is αn and the number of samples in $S^2_{\texttt{val}}$ is $(1 - \alpha)n$, where $\alpha \in [0, 1]$. After the split you are going to follow the procedure in Point 3. What would be your considerations for selecting α? Specifically:

   (a) What is good and what is bad about having large α (close to 1, so that $S^1_{\texttt{val}}$ is large and $S^2_{\texttt{val}}$ is small) and what is good and what is bad about having small α (close to 0)?

   (b) How would you select α (as a function of M and δ)? [There is no single right answer to this question, but what considerations would you take into account and what value would you use?]

2 Occam's Razor

We want to design an application for bilingual users. The application should detect the language in which the person is typing based on the first few letters typed. In other words, we want to design a classifier that takes a short string (that may be less than a full word) as input and predicts one of two languages, say, Danish or English. For simplicity we will assume that the alphabet is restricted to a set Σ of 26 letters of the Latin alphabet plus the white space symbol (so in total $|\Sigma| = 27$). Let Σ^d be the space of strings of length d. Let $\mathcal{H}_d$ be the space of functions from Σ^d to $\{0, 1\}$, where Σ^d is the input string and $\{0, 1\}$ is the prediction (Danish or English). Let $\mathcal{H} = \bigcup_{d=0}^{\infty} \mathcal{H}_d$ be the union of $\mathcal{H}_d$-s.

1. Derive a high-probability bound[1] for $L(h)$ that holds for all $h \in \mathcal{H}_d$.
2. Derive a high-probability bound for $L(h)$ that holds for all $h \in \mathcal{H}$.
3. Explain the trade-off between picking short strings (small d) and long strings (large d). Which terms in the bound favor small d (i.e., they increase with d) and which terms in the bound favor large d (i.e., they decrease with d)?

Optional, not for submission You are very welcome to experiment with the influence of the string length d on the performance. You can find a lot of texts in different languages here `http://www.gutenberg.org/catalog/`. Do you observe the effect of initial improvement followed by overfitting as you increase d?

3 Kernels

The first question should improve the understanding of the geometry of the kernel-induced feature space. You can directly use the result to implement a kernel nearest-neighbor algorithm.

The second question should make you more familiar wth the basic definition of the important concept of positive definiteness.

The third question is important to understand the real dimensionality of learning problems using a linear kernel – one reason why linear kernels are often treated differently in efficient implementations.

[1] A bound that holds with probability at least $1 - \delta$.

3.1 Distance in feature space

Given a kernel k on input space $\mathcal{X}$ defining RKHS $\mathcal{H}$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ denote the corresponding feature map (think of $\Phi(x) = k(x, .)$). Let $x, z \in \mathcal{X}$. Show that the distance of $\Phi(x)$ and $\Phi(z)$ in $\mathcal{H}$ is given by

$$\|\Phi(x) - \Phi(z)\| = \sqrt{k(x, x) - 2k(x, z) + k(z, z)}$$

(if distance is measured by the canonical metric induced by k).

3.2 Sum of kernels

Let $k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be positive-definite kernels.

Proof that $k(x, z) = k_1(x, z) + k_2(x, z)$ is also positive-definite.

3.3 Rank of Gram matrix

Let the input space be $\mathcal{X} = \mathbb{R}^d$. Assume a linear kernel, $k(x, z) = x^{\mathrm{T}} z$ for $x, z \in \mathbb{R}^d$ (i.e., the feature map Φ is the identity) and m input patterns $x_1, \ldots, x_m \in \mathbb{R}^d$.

Proof an upper bound on the rank of the Gram matrix from the m input patterns in terms of d and m.