

Machine Learning, Spring 2018: Exercise Sheet 9 - Solutions

Problem 1. Consider the polynomial curve fitting example from Section 7.2.1 in the LNs. In the LNs I say that polynomial curve fitting can be done by calling a ready-made function *polyfit* that Matlab offers you. In fact, *polyfit* is just a special case of linear regression. Find out for yourself how *polyfit* can be programmed (without peeking at the Matlab documentation). More specifically, consider the following optimization problem. Given: a sample $(x_i, y_i)_{i=1, \dots, N}$ of argument-value training points, where x_i, y_i are real numbers. Also given: a maximal polynomial degree m . Design an algorithm that finds the optimal weight vector $W = (w_0, \dots, w_m)'$ defined by the loss function

$$L(W) = \frac{1}{N} \sum_{i=1}^N (p(x_i) - y_i)^2$$

All you have to do is to find a way to translate this problem into an instance of the linear regression problem specified in Equation (17) in the LNs.

Problem 2. Let $\gamma[\mu, \sigma^2]$ denote the pdf of a normal distribution with expectation μ and variance σ^2 . A real-valued random variable X has a *mixture of Gaussians* (MoG) pdf $p(x) = \sum_{i=1, \dots, k} \alpha_i \gamma[\mu_i, \sigma_i^2](x)$, where the k *mixture coefficients* α_i are non-negative and sum to 1. What is $E[X]$? – *Note:* Mixture of Gaussians representations of distributions are very popular in ML because they combine simplicity with flexibility. The graphics in Figure 23 in the LN actually show a MoG example.

Problem 3. Given a size- N training sample $(x_i, y_i)_{i=1, \dots, N}$ of argument-value training points, setting up an L -fold cross validation scheme for model optimization requires one to decide on the number L of folds in the first place. The two extreme cases are $L = 2$ (smallest possible number of folds) and $L = N$ (largest possible number of folds, "leave-one-out cross-validation"). Following the routines of the cross-validation procedure, in each of these cases one will determine an optimal model capacity. Here is a claim:

"When one uses 2-fold cross-validation, the optimal model capacity found is likely to be too low, that is, one will settle on a model capacity $m_{opt \text{ 2-fold}}$ which is smaller than the true optimal model capacity m_{opt} and hence will give underfitting models in the end; whereas if one uses leave-one-out cross-validation, one will determine a model capacity $m_{opt \text{ N-fold}}$ which is greater than m_{opt} , hence ultimately leading to overfitting."

Is this claim true? Give your reasoning in plain English.

Solution to problem 1. Using the notation from Eqn (17) in the LN, for $i = 1, \dots, N$ define the size- $(m+1)$ feature vector $\mathbf{f}_i = (x_i^0, x_i^1, \dots, x_i^m)'$, and put $z_i = y_i$ – and you are done!

Solution to problem 2. $E[X] = \sum_{i=1, \dots, K} \alpha_i \mu_i$

Solution to problem 3. The truth is that whenever one uses cross-validation to determine the optimal model capacity m_{opt} , one obtains a result that under-estimates the correct value – that is, $m_{opt, L-fold} < m_{opt, true}$ – regardless of L . As a consequence, when one uses the found $m_{opt, L-fold}$ for learning the final model on the entire training data set, one under-fits. The reason is that with L -fold cross validation, the obtained model capacity $m_{opt, L-fold}$ is optimal for a training data set of size $N - L$, not for a size N dataset. But the smaller the data set size, the more susceptible model learning becomes for overfitting; that is, the more strongly will the cross-validation "attempt" to compensate for overfitting. This effect of systematic under-estimation becomes smallest for leave-one-out cross-validation, because $N - 1$ is approximately equal to N . Therefore, for high-quality model estimates (trying to get models that best generalize to new test data), leave-one-out cross-validation is the best one can do – but it is also the most expensive way to go. In summary, one will opt for the largest number of folds that one can afford with regards to computational effort.