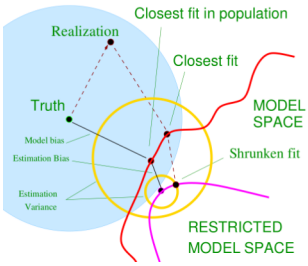
Solution exam 2018

Exercise 1: Penalized regression

1a)



Using the figure above, explain the concept of bias-variance trade-off
ans: When using the squared-error loss, the expected prediction error can be decomposed into three parts:

- an irreducible error, which cannot be avoided;
- the squared bias, where bias is the difference between the average of the estimate and the true mean;
- a variance term, the expected square deviation of our estimate from its mean

When minimizing the expected prediction error, we are in a situation in which if we reduce the variance component, for example by adding constraints to the model space, we increase the squared bias, and vice versa. This is called bias-variance trade-off. In the picture this is represented by lines and circles: when adding constraints, o.e. when we move from the model space to the restricted model space, we increase the bias (represented by the line named "Estimation Bias") but we reduce the variance (the radius of the circle around the "Closest fit in population" point is smaller than that around the corresponding point on the restricted model space
1b) Show analytically the same concept of the point above by mathematicallycomparing bias and variance of the ordinary least square estimator and ofthe ridge estimator
ans: bias:

$$E[\hat{\beta}_{OLS}] = \beta \tag{1}$$
$$E[\hat{\beta}_{ridge}] = E[(X^TX + \lambda I)^{-1}X^Ty] \tag{2}$$
$$= E[(\lambda I + \lambda(X^TX)^{-1})^{-1}(X^TX)^{-1}X^Ty] \tag{3}$$
$$= E[(\lambda I + \lambda(X^TX)^{-1})^{-1}]E[\hat{\beta}_{OLS}] \tag{4}$$
$$= w_\lambda \beta \Rightarrow E[\hat{\beta}_{ridge}] \neq \beta \text{ for } \lambda > 0 \tag{5}$$

variance:

$$Var[\hat{\beta}_{OLS}] = \sigma^2(X^TX)^{-1} \tag{6}$$
$$Var[\hat{\beta}_{ridge}] = Var[w_\lambda \hat{\beta}_{OLS}] \tag{7}$$
$$= w_\lambda Var[\hat{\beta}_{OLS}]w_\lambda^T \tag{8}$$
$$= \sigma^2 w_\lambda (X^TX)^{-1} w_\lambda^T. \tag{9}$$

then:

$$Var[\hat{\beta}_{OLS}] - Var[\hat{\beta}_{ridge}] = \sigma^2[(X^TX)^{-1} - w_\lambda(X^TX)^{-1}w_\lambda^T] \tag{10}$$
$$= \sigma^2 w_\lambda[(I + \lambda(X^TX)^{-1})(X^TX)^{-1}(I + \lambda(X^TX)^{-1})^T - (X^TX)^{-1}]w_\lambda^T \tag{11}$$
$$= \sigma^2 w_\lambda[((X^TX)^{-1} + 2\lambda(X^TX)^{-2} + \lambda^2(X^TX)^{-3}) - (X^TX)^{-1}]w_\lambda^T \tag{12}$$
$$= \sigma^2 w_\lambda[2\lambda(X^TX)^{-2} + \lambda^2(X^TX)^{-3}]w_\lambda^T > 0 \tag{13}$$

(since all terms are quadratic and therefore positive)

$$\Rightarrow Var[\hat{\beta}_{ridge}] \leq Var[\hat{\beta}_{OLS}] \tag{14}$$

Exercise 2: Bootstrapping for model evaluation

2) Bootstrapping for model selection. Consider the following procedure to estimate the prediction error:

1. generate B bootstrap samples $z_1, \ldots, z_B$, where $z_b = (y_1^*, x_1^*), \ldots, (y_N^*, x_N^*)$, $b = 1, \ldots, B$, and (y_i^*, x_i^*), $i = 1, \ldots, N$ is an observation sampled from the original dataset;
2. apply the prediction rule to each bootstrap sample to derive the predictions $\hat{f}_b^*(x_i)$, $b = 1, \ldots, B$;
3. compute the error for each point, and take the average:

$$\hat{\text{Err}}_{\text{boot}} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}_b^*(x_i)) \tag{15}$$

2a) Explain why this procedure is incorrect and suggest a different way to proceed which still uses a bootstrap approach.
ans: The procedure is incorrect because the prediction error is computed on observations already used to train the prediction rule. This leads to underestimation of the error (writing "too optimistic" was acceptable).A possible solution is to compute the prediction error only on those observations (in average 36.8% of the original sample) not included in the bootstrap sample. Since this approach leads to overestimating the prediction error, solutions like those described in the point (b) has been implemented
2b) Describe the 0.632 bootstrap and the 0.632+ bootstrap procedures, explaining in particular the rationale behind their construction.
ans: The 0.632 bootstrap procedure addresses the problem of overestimation of the correct procedure described in point (a) by averaging it (with weight 0.632 and 0.368, respectively) with the training error(underestimated error). The result is a sort of compromise between overestimation and underestimation. In formula:

$$\hat{\text{Err}}^{(0.632)} = 0.632\hat{\text{Err}}^{(1)} + 0.368\bar{\text{err}}, \tag{16}$$

where $\bar{\text{err}}$ is the training error and $\hat{\text{Err}}^{(1)}$ is the corrected procedure described at point (a). Since the 0.632 and 0.382 weights may not be the best choice (e.g., in case of complete overfitting in the training set), the 0.632+ bootstrap has been developed. In the latter procedure

$$\hat{\text{Err}}^{(0.632+)} = \hat{w}\hat{\text{Err}}^{(1)} + (1 - \hat{w})\bar{\text{err}} \tag{17}$$

the weights depend on the relative overfitting rate, so the 0.632+ bootstrap can be seen as a better compromise between the overestimation and underestimation of the prediction error done by the corrected procedure described at point (a) and the training error, respectively.

Exercise 3: Smoothing Splines

Consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimizes the penalized residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{N}(y_i - f(x_i))^2 + \lambda\int\{f''(t)\}^2 dt$$

where $\lambda \geq 0$.

3a) Define the role of the penalization term $\lambda\int\{f''(t)\}^2 dt$ in relation to its specific form, and discuss what happens when the smoothing parameter λ varies.
ans: The penalization term penalizes curves too "wiggly", reducing the model complexity by penalizing curves with high curvature. The amount of penalty is controlled by the tuning parameter λ:

- when $\lambda = 0$ there is no penalization, and it leads to a curve which passes though all the points
- when $\lambda = \infty$ no curvature is allowed, and it leads to a straight line.

The choice of λ is also a case of bias-variance trade-off: smaller λ, smaller the bias (and higher the variance); larger λ, larger the bias (and smaller the variance).
3b) The solution of minimization problem (1) the can be written as a natural spline.

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j. \tag{18}$$

Rewrite (1) as a function of θ (i.e.,$RSS(\theta, \lambda)$) and use its solution to show that a smoothing spline for a fixed λ is a linear smoother (linear operator). Use it to define the effective degrees of freedom of a smoothing spline.
ans: Rewriting equation 18 in terms of θ, i.e. by plugging in $f(x) = \sum_{i=1} N_i(x)\theta_i$, one obtains the form

$$RSS(\theta, \lambda) = (y - N\theta)^T(y - N\theta) + \lambda\theta^T\Omega_N\theta, \tag{19}$$

where $\{N\}_{ij} = \{N_j(x_i)\}$ and $\{\Omega_N\}_{jk} = \int N''(t)N''(t)dt$. Either deriving (and setting the first derivative equal to 0) or recognizing the solution of a regularized ridge regression, one obtains

$$\theta = (N^TN + \lambda\Omega_N)^{-1}N^Ty. \tag{20}$$

Therefore

$$\hat{f} = N(N^TN + \lambda\Omega_N)N^Ty, \tag{21}$$

which is linear in y. Knowing that the degrees of freedom of a linearsmoother correspond to the trace of the smoothing matrix,

$$dof(\hat{f}) = \text{trace}(N(N^TN + \lambda\Omega_N)N^T). \tag{22}$$

Exercise 4: Bagging

4a) Describe bagging, mentioning at least one advantage with respect to a singletree and a disadvantage with respect to a boosted tree model
ans: Bagging (Bootstrap AGGregatING) is a procedure which consists in aggregating (by averaging, by voting, etc.) the results of a prediction rule applied to a number of bootstrap samples generated from the original data. The prediction rule is typically (but not necessarily) a tree.
An advantage of bagging with respect to a single tree is its stability (reduced variance), while in contrast to a boosted tree model bagging is not able to take advantage of the results of the previous iterations to improve the later predictions (e.g., in the context of classification, AdaBoost is able to focus on misclassified observations by weighting them more in the later iterations).
4b) Consider a classification problem and how to aggregate the results of the single trees in a bagging classifier. The aggregation can be done by looking at the estimated classes or at the class-probability estimates. Show with a simple example that the two procedures can produce different results in terms of classification of an observation
ans: Consider a simple case of binary classification in which we aggregate the results of three trees: two trees say that an observation x_i is of class A with probability 0.55, of class B with probability 0.45; the third tree,instead, says A with probability 0.1, B with probability 0.9
When aggregating by "majority of votes",xiis classified as A (twovotes against one).
When aggregating by "class probabilities",x_i is classified as B (average probabilities being 0.4 for A, 0.6 for B).

Exercise 5: Boosting

5a) Show that the additive expansion produced by AdaBoost is estimating one-half the log-odds of $P(Y = 1|X = x)$, where Y is the binary response and X is the input matrix.
ans: Following is the solution to exercise 10.2 in the textbook:

$$f^*(x) = \text{argmin}_{f(x)}E_{Y|X=x}[e^{-Yf(x)}] \tag{23}$$
$$\frac{\partial E_{Y|X=x}[e^{-Yf(x)}]}{\partial f(x)} = E_{Y|X=x}[-Ye^{-Yf(x)}] \tag{24}$$
$$E_{Y|X=x}[-Ye^{-Yf(x)}] = 0 \Rightarrow Y = \begin{cases} -1 & \text{for } Pr[Y=-1|X=x] \\ 1 & \text{for } Pr[Y=1|X=x] \end{cases} \tag{25}$$

Inserting:

$$-(-1)e^{-(-1)f(x)}Pr[Y=-1|X=x] \cdot -(1)e^{-(1)f(x)}Pr[Y=1|X=x] = 0 \tag{26}$$

Multiplying with $e^{f(x)}$ on both sides yields:

$$e^{2f(x)}Pr[Y=-1|X=x] = Pr[Y=1|X=x] \tag{27}$$
$$e^{2f(x)} = \frac{Pr[Y=1|X=x]}{Pr[Y=-1|X=x]} \tag{28}$$
$$f(x) = \frac{1}{2}\log\left(\frac{Pr[Y=1|X=x]}{Pr[Y=-1|X=x]}\right) \tag{29}$$

5b) Consider the following algorithm:

1. initialize the estimate, e.g. $f_0(x) = 0$
2. for $m = 1, \ldots, m_{stop}$:
   (a) compute the negative gradient vector, $u_m =$ **OM**
   (b) fit the base learner to the negative gradient vector, $h_m(u_m, x)$.
   (c) update the estimate $f_m(x) = f_{m-1}(x) + \nu h_m(u_m, x)$.
3. final estimate is then:

$$\hat{f}_{m_{stop}}(x) = \sum_{m=1}^{m_{stop}} \nu h_m(u_m, x) \tag{30}$$

Name the specific boosting algorithm and write the complete formula in place of **OM**, for a generic loss function $L(y, f(x))$.
ans: The algorithm is called '**Gradient Boosting**', and the omitted expression is:

$$u_m = -\frac{\partial L(y, f(x))}{\partial f(x)}\Big|_{f(x)=f_{m-1}(x)} \tag{31}$$