

UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Exam in: STK-IN4300/9300 — Statistical Learning
Methods in Data Science

Day of examination: Wednesday, December 10, 2025

Examination hours: 9.00–13.00

This solution proposal consists of 5 pages.

Appendices: None.

Permitted aids: None.

Solution Proposal

Problem 1 Bias-Variance Tradeoff

- (a) The bias-variance tradeoff describes the relationship between two types of errors that control how well a supervised learning method will generalize to new test data (test error):
- **Bias:** Systematic error due to wrong (or too strong) assumptions in the learning algorithm;
 - **Variance:** Random error that is due to a sensitivity to fluctuations in the training set.

As seen in the figure, the bias-variance tradeoff is connected to the complexity of the model, such that too high complexity leads to overfitting (low bias, high variance) and too low complexity leads to underfitting (high bias, low variance). Overfitting is associated with a low training error but high test error and underfitting is associated with a high training and test error. One wants to find a model with optimal complexity that balances bias and variance to result in the lowest possible test error.

- (b) (i) The first term of the regularized RSS criterion measures how well the function fits the data and the second term penalizes curvature of the function by integrating over the squared second derivative of the function, effectively penalizing “wiggly” behavior. Thus, λ is a smoothing parameter that controls the smoothness (or complexity) of the function: increasing the value of λ increases the bias, but decreases the variance, and vice versa.
- (ii) When $\lambda \rightarrow 0$, $\hat{f}$ can be any function that interpolates the data. When $\lambda \rightarrow \infty$, the model will be reduced to the simple least squares fit, as no non-zero second derivatives are allowed.

(Continued on page 2.)

- (c) (i) For a target input point x_0 , KNN uses the observations in the training data that are closest to x_0 in the input space, w.r.t. some specified distance, to form a prediction. Specifically, in the case of regression, the predicted value at x_0 is given by

$$\hat{y}_0 = \frac{1}{k} \sum_{x_i \in N_k(x_0)} y_i,$$

where $N_k(x_0)$ is the neighborhood of x_0 defined by the k closest observations in the training data. In the case of classification, the prediction is typically given by the majority vote in the neighborhood, that is, y_0 is assigned the most frequent class label among the training observations in the neighborhood.

- (ii) The bias-variance tradeoff is directly controlled by k , that is, the size of neighborhood (which is inversely related to model complexity). Decreasing k gives a reduced bias, as the predictions are based on observations closer to the target point, while the variance is increased, as the prediction is based on fewer data points (and vice versa).

Problem 2 Linear Regression

- (a) (i) $\hat{\theta}$ is a linear estimator, since:

$$\hat{\theta} = x_0^T \hat{\beta} = x_0^T \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T}_{=\hat{\mathbf{c}}^T} \mathbf{y} = \hat{\mathbf{c}}^T \mathbf{y}.$$

- (ii) $\hat{\theta}$ is an unbiased estimator, since:

$$\mathbb{E}[\hat{\theta}] = \mathbb{E}[x_0^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}] = x_0^T (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{X}) \beta = x_0^T \beta = \theta,$$

where the second equality follows from the fact that $\mathbb{E}[\mathbf{y}] = \mathbf{X}\beta$.

- (b) (i) The second term of the EPE (squared bias) disappears for an unbiased estimator and the first terms (irreducible error) cannot be decreased. That leaves us with the third term (variance), which by the Gauss-Markov theorem is minimized by the least squares estimator. That is, among the class of linear unbiased estimators, the least squares estimator is optimal in terms of the EPE (under the given assumptions).
- (ii) In practice, one might (and often will) get a smaller EPE by introducing some bias (e.g., shrinkage) in order to reduce the variance even more (bias-variance tradeoff).
- (c) Best subset selection (subset size M) drops all variables (i.e., sets the coefficients to zero) for which the least squares coefficients are smaller than the M :th largest. Ridge regression does a proportional shrinkage, shrinking larger coefficients more such that the amount of shrinkage is reduced linearly as the coefficients gets closer to zero (no variable selection). Lasso shrinks each coefficient by a constant factor, λ , and sets the coefficient to zero if its absolute value is smaller than λ (variable selection).

(Continued on page 3.)

Problem 3 Effective Number of Parameters

- (a) The KNN prediction of observation i can be expressed as

$$\hat{y}_i = \sum_{j=1}^N \frac{1}{k} I(x_j \in N_k(x_i)) \cdot y_j,$$

which can be equivalently expressed using vector notation:

$$\hat{y}_i = \underbrace{\left(\frac{1}{k} I(x_1 \in N_k(x_i)), \dots, \frac{1}{k} I(x_N \in N_k(x_i)) \right)}_{=S_i}^T \mathbf{y}.$$

Letting S_i denote the i :th row of $\mathbf{S}$, we can thus express the KNN prediction vector as:

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y},$$

where $\mathbf{S}$ depends only on the inputs, implying that KNN regression is a linear fitting method.

- (b) Since KNN regression is a linear fitting method, we can define the effective number of parameters as

$$\text{df}(\mathbf{S}) = \text{trace}(\mathbf{S}) = \sum_{i=1}^N \frac{1}{k} I(x_i \in N_k(x_i)) = \frac{N}{k},$$

since each training observation is (typically) included in its own neighborhood. For $k = 1$, we get N parameters (one per training observation) and for $k = N$ we get 1 parameter (one constant value given by the global mean).

Problem 4 Model Selection and Evaluation

- (a) (i) This is the generalization error or expected test error when we when we compute the expectation over the test point and the training data, rather than keeping the latter fixed. This error can be estimated by cross-validation or bootstrap procedures (Note: it is enough to mention an estimation procedure without describing it).
- (ii) This is the in-sample error, which is the average expected prediction error over the training sample. Specifically, we keep the input values and compute the expected error assuming that only the output values are redrawn, meaning that the test and training inputs coincide. Model selection criteria such as the C_p statistic and AIC can be viewed as estimates of the in-sample error, and arguably also BIC even though it is motivated primarily through the Bayesian approach to model selection (Note: again, it is enough to mention an estimation procedure).

(Continued on page 4.)

(b) We start by noting that

$$\begin{aligned}
 \mathbb{E}_{X,Y}[R_{te}(\hat{\beta})] &= \mathbb{E}_{X,Y}\left[\frac{1}{M} \sum_{i=1}^M (\tilde{Y}_i - \hat{\beta}^T \tilde{X}_i)^2\right] \\
 &= \frac{1}{M} \sum_{i=1}^M \mathbb{E}_{X,Y}[(\tilde{Y}_i - \hat{\beta}^T \tilde{X}_i)^2] \\
 &= \mathbb{E}_{X,Y}[(\tilde{Y} - \hat{\beta}^T \tilde{X})^2] \\
 &= \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{X,Y}[(\tilde{Y}_i - \hat{\beta}^T \tilde{X}_i)^2],
 \end{aligned}$$

meaning that the average expected error on the test data is independent of the number of test points, and can thus be set to match the number of training observations, N . Then, since

$$R_{te}(\hat{\beta}) \geq R_{te}(\beta^*), \text{ where } \beta^* = \arg \min_{\beta} R_{te}(\beta)$$

we have that (by the monotonicity property of expectation):

$$\mathbb{E}_{X,Y}[R_{te}(\hat{\beta})] \geq \mathbb{E}_{X,Y}[R_{te}(\beta^*)].$$

That is, we have that:

$$\begin{aligned}
 \mathbb{E}_{X,Y}[R_{te}(\hat{\beta})] &= \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{X,Y}[(\tilde{Y}_i - \hat{\beta}^T \tilde{X}_i)^2] \\
 &\geq \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{X,Y}[(\tilde{Y}_i - \beta^{*T} \tilde{X}_i)^2] \\
 &= \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{X,Y}[(Y_i - \hat{\beta}^T X_i)^2] \\
 &= \mathbb{E}_{X,Y}[R_{tr}(\hat{\beta})].
 \end{aligned}$$

Problem 5 Gradient Tree Boosting

(a) The gradient tree boosting procedure performs the following steps

Step 1. Initializes with the optimal constant model (which corresponds to a single terminal node tree).

Step 2. At each iteration, $m = 1 \dots, M$:

- (a) For each training observation, $i = 1, \dots, N$: compute the negative partial derivative r_{im} of the loss function with respect to the function $f(x_i)$ evaluated at $f(x_i) = f_{m-1}(x_i)$.
- (b) Fit a regression tree with the negative partial derivatives computed in (a) as target using the squared error loss function (i.e., not the target loss), resulting in a tree with terminal regions R_{jm} , $j = 1, \dots, J_m$ (representing a partition of the input space).

(Continued on page 5.)

- (c) For each terminal region $j = 1, \dots, J_m$: compute the optimal constant value, γ_{jm} , that when added to the function output minimizes the target loss function $L(\cdot, \cdot)$ w.r.t. the training observations in terminal region R_{jm} .
 - (d) Update the current function $f_{m-1}(x)$ by adding a region-specific constant γ_{jm} to the output of all input configurations associated with R_{jm} .
- Step 3. After completing the M :th iteration, output the final model $f_M(x)$.
- (b) (i) M denotes the number of steps, or boosting iterations, and it can be used to prevent overfitting by stopping the boosting procedure before reaching the optimum of the loss (early stopping).
 J controls the size of the individual trees, which has a direct effect on the expressiveness of the model. Specifically, by using trees of size J the model can have interactions that are at most of level $J - 1$.
- (ii) If we set $J = 1$, we will only get a single terminal leaf in each tree, and the model will thus be reduced to a global constant that will depend on the loss function (for example, the average value in case of a squared error loss). In other words, the algorithm will not update the model after Step 1.

THE END!