Statistical Learning Exam 2018

9 January 2019, 14:00–17:00

- You are allowed to use the book, your private notes and any other printed or written materials.
- Every subquestion (1a, 1b, etc.) counts for the same number of points.

1. **[Asymmetric Classification Loss]** Consider binary classification with class -1 occurring much less frequently than class $+1$. Suppose we are especially interested in class -1, so we want to modify the standard 0/1-loss for binary classification such that a failure to recognize class -1 counts for 10 times as much loss as a failure to recognize class $+1$.

   (a) Give the values of the new loss $L(y, \hat{y})$ for all combinations of y and $\hat{y}$.
   ANSWER: Accept any answer such that $L(-1, +1) = 10L(+1, -1)$ and $L(y, y) = 0$.

   (b) What is the Bayes optimal classifier for the new loss?
   ANSWER: Suppose $(\boldsymbol{x}, y) \sim P^*$. Then the Bayes optimal classifier f_B satisfies

   $$f_B(\boldsymbol{x}) = \begin{cases} +1 & \text{if } P^*(y = 1 \mid \boldsymbol{x}) \geq \frac{10}{11}, \\ -1 & \text{otherwise,} \end{cases}$$

   or, equivalently,

   $$f_B(\boldsymbol{X}) = \begin{cases} +1 & \text{if } P^*(y = +1 \mid \boldsymbol{x}) \geq 10P^*(y = -1 \mid \boldsymbol{x}), \\ -1 & \text{otherwise.} \end{cases}$$

   It is also correct to classify as -1 in case of ties, i.e. $P^*(y = 1 \mid \boldsymbol{x}) = \frac{10}{11}$.

   (c) Naive Bayes implicitly assumes the loss is the 0/1-loss. How should it be modified for the new loss?
   ANSWER: Let $\hat{P}$ be the Naive Bayes estimate of P^*. Then classify as in the previous question, except that P^* is replaced by $\hat{P}$.

2. **[Constructing Features]** Consider the regression tree in Figure 1, which takes a feature vector $\boldsymbol{x} = (x_1, x_2, x_3)^\top$ as input. By changing the values $v_1, \ldots, v_5$ in the leaves, this tree can represent a class of regression functions. Construct features such that a linear regression model with your features can represent exactly the same class of regression functions as this tree.

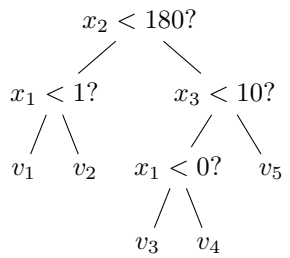


Figure 1: Decision Tree

ANSWER:

$$\tilde{x}_1 = I(x_2 \geq 180 \text{ and } x_1 \geq 1)$$
$$\tilde{x}_2 = I(x_2 \geq 180 \text{ and } x_1 < 1)$$
$$\tilde{x}_3 = I(x_2 < 180 \text{ and } x_3 \geq 10 \text{ and } x_1 \geq 0)$$
$$\tilde{x}_4 = I(x_2 < 180 \text{ and } x_3 \geq 10 \text{ and } x_1 < 0)$$
$$\tilde{x}_5 = I(x_2 < 180 \text{ and } x_3 < 10)$$

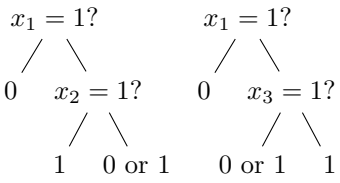
3. [**Decision Trees**] Consider the following data:

y	1	1	0	0
	1	1	1	0
$\boldsymbol{x}$	1	0	1	0
	1	0	0	1

(a) Grow a classification tree using the Gini index with CART and stop splitting at depth 2 or when the remaining examples are all from the same class. You may break ties anyway you want. Do not do any pruning. Draw the resulting tree and give its training error.
ANSWER: It is not necessary to describe the details of the CART procedure, but they are:

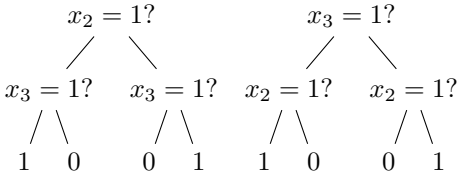
```
 CART:
 SPLIT 1 (on feature 1):
   split on feature 1: Gini index gives 1 outcome * 0 + 3 outcomes * 2*2/3*1/3 = 12/9
   split on feature 2: Gini index gives 2 * 2 outcomes * (2 * 1/2 * 1/2) = 2
   split on feature 3: Gini index gives 2 * 2 outcomes * (2 * 1/2 * 1/2) = 2
 SPLIT 2 for group:
((1, 1, 1), 1)
((1, 0, 0), 1)
((1, 1, 0), 0)
   split on feature 2: Gini index gives 2 outcomes * 2*1/2*1/2 + 1 outcomes * 0 = 1
   split on feature 3: Gini index gives 2 outcomes * 2*1/2*1/2 + 1 outcomes * 0 = 1
 No more splits for group:
((0, 0, 1), 0)
```

Thus any of the following trees are correct, depending on how you break ties:



The corresponding training error is **25%** in all cases.

(b) Does there exist a decision tree of depth at most 2 with smaller training error? If your answer is no, explain why not. If your answer is yes, give the smallest possible training error and draw a tree that achieves it.
ANSWER: Yes, either one of the following trees achieves 0% training error:



4. [**Cross-validation**] Consider a trivial classifier that does not look at the data, but always outputs the same classification function. What would the 5-fold cross-validation error of this estimator be

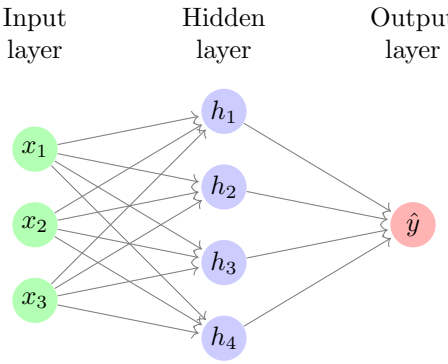


Figure 2: A neural network

on a data set of size 1000? Would it be a good estimate of the expected prediction error (EPE) of the classifier?
ANSWER: In this case, the cross-validation error is equal to the training error for the function. This would be a good estimate, because the function does not depend on the data.

5. **[Neural Networks]**

   (a) Suppose we have a neural network[1] that takes as input a 100-dimensional feature vector. Then it has two hidden layers with 200 hidden units each. And finally it produces a single output. If we double the sizes of the two hidden layers, how much longer will it roughly take to compute a single gradient with backpropagation? Give your answer either as a factor that is correct up to one decimal place of precision, or as a fraction. To simplify your calculations, pretend that the network does not have any intercepts, because the answer will be of the same order of magnitude anyway.
   ANSWER: Original network roughly has: 100 x 200 + 200 x 200 + 200 x 1 = 60,200 parameters. New network roughly has: 100 x 400 + 400 x 400 + 400 x 1 = 200,400 parameters. This is roughly a factor of 3.3 more parameters. As the runtime of backpropagation grows linearly with the number of parameters, we expect it to take roughly 3.3 times as long.

   (b) A neural network with a single hidden layer and one output can be expressed as a linear combination of basis functions

$$f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m b_m(\boldsymbol{x}),$$

   where the basis functions $b_1, \ldots, b_m$ are not fixed, but depend on the parameters of the network. What are the basis functions b_m and coefficients β_m (including β_0) that correspond to the neural network in Figure 2? In your answer, use the following symbols:

   - $w_{i,j}$: the weight between x_i and h_j
   - $w_{0,j}$: the intercept for h_j
   - v_j: the weight between h_j and $\hat{y}$
   - v_0: the intercept for $\hat{y}$
   - σ: a fixed activation function.

   ANSWER: For $j = 1, \ldots, 4$, $b_j(\boldsymbol{x}) = \sigma(w_{0,j} + \sum_{i=1}^{3} w_{i,j} x_i)$. Then let $\beta_0 = v_0$, $\beta_m = v_m$ for $m = 1, \ldots, 4$.

6. **[Clustering]** Consider clustering with the K-means algorithm.

[1] Assume a basic fully connected network without any special layers like convolutional layers.

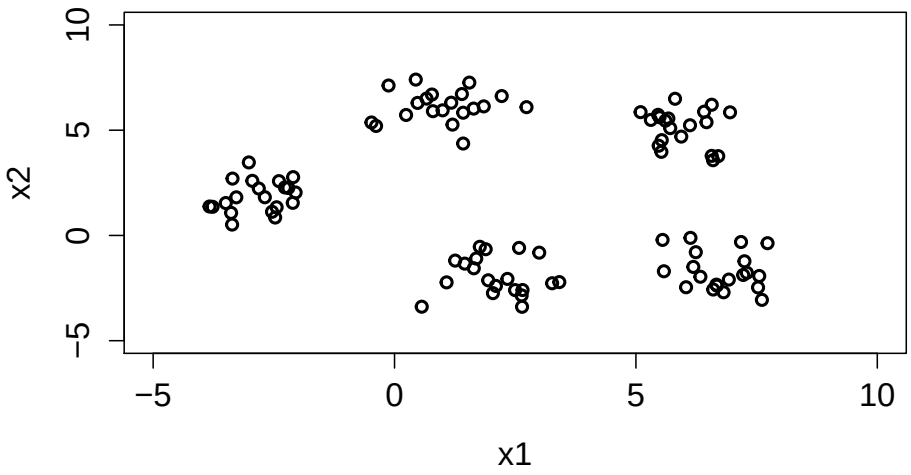


Figure 3: Unsupervised data with two features without any labels

(a) If we choose the number of clusters K to be between 1 and 10, which value for K do you expect to give the smallest sum of squared distances

$$\sum_{k=1}^{K} \sum_{i:C(i)=k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{1}$$

for the data in Figure 3 after running K-means? Here $C(i)$ is the cluster assigned to data point $\boldsymbol{x}_i$, and $\boldsymbol{\mu}_k$ is the mean of the k-th cluster. NB Don't forget that the book contains a mistake in the definition of K-means: K-means aims to minimize (1), not (14.31) from the book.
ANSWER: $K = 10$, because the more clusters, the better we can fit the data.

(b) Does K-means always give a stable solution in the following sense? Suppose we run K-means on a data set until convergence. Then we move one of the data points a tiny bit, and we continue running K-means until it converges again. We consider the algorithm stable if the cluster means only change a tiny bit between the two cases. If yes, provide arguments supporting your statement. If no, provide a counterexample.
ANSWER: No: counterexample is two clusters with not too many points that are very far away, with one point approximately in the middle between the two. Depending on which cluster this point is assigned to, the mean for that cluster will move a lot. So by moving this point a tiny bit, we can significantly change the cluster means. NB Students would be expected to draw a picture of the counterexample.

7. **[Boosting]** AdaBoost may be interpreted as forward stagewise additive modeling (FSAM) with the exponential loss. FSAM computes updates of the form

$$(\beta_m, \gamma_m) \;=\; \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L\big(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta b(\boldsymbol{x}_i;\gamma)\big),$$

where $b(\boldsymbol{x}_i;\gamma)$ is the prediction of the weak learner with parameters γ.

(a) Consider an example $(\boldsymbol{x}_i, y_i)$ for which $f_{m-1}(\boldsymbol{x}_i) + \beta_m b(\boldsymbol{x}_i;\gamma_m)$ is very far from 0, but gives the wrong classification. Would the loss of this example in the FSAM update be much smaller, much larger or approximately the same if we would replace the exponential loss by the hinge

loss with the same parameters β_m, γ_m?
ANSWER: Much smaller, because the margin of is very negative and the hinge loss is much smaller than the exponential loss for very negative margin.

(b) Answer the same question for an example $(\boldsymbol{x}_i, y_i)$ for which $f_{m-1}(\boldsymbol{x}_i) + \beta_m b(\boldsymbol{x}_i; \gamma_m)$ is very far from 0 and gives the *correct* classification.
ANSWER: Approximately the same, because the margin is very positive and both losses are (approximately) zero for large positive margin.

(c) It is a well-known phenomenon that the test error of AdaBoost can keep decreasing if we do more rounds of boosting even after the training error is already zero. This happens, for instance, for decision stumps. Would it also happen if we use a deep neural network as our weak learner? Assume that the network is applied to a complicated image classification task where it is able to achieve zero training error but its test set error (without boosting) is 25%. Explain your answer.
ANSWER: Boosting will not help. Possible motivations:

i. If the training error of the weak learner is 0, then the output of AdaBoost will be identical to not doing any boosting at all. (To see this, notice that the weights w_i in AdaBoost will not change if the weak learner makes no mistakes. Consequently $G_m = G_{m-1} = \cdots = G_1$ for all rounds m.)

ii. Boosting only helps to reduce bias; it is not intended to reduce variance. The neural network has no problems with bias, because it is already able to achieve 0 training error. Thus the neural network is actually already a strong learner, for which boosting is not appropriate.