Machine Learning (in Finance)
FEM31002 & FEM21045

Sample exam with solutions

September 28, 2020

1) Answer the following questions for the k-nearest neighbor (k-NN) algorithm.

a. Explain how the selection of k for a k-NN classifier affects the model's flexibility, variance and bias.

Solution: When $k = 1$, the decision boundary is overly flexible. This corresponds to a classifier that has low bias but very high variance. As k grows, the method becomes less flexible and produces a decision boundary that is close to linear. This corresponds to a low-variance but high-bias classifier.

b. Assume that your data can be classified with a linear Bayes decision boundary. Which value of k gives better results? Why?

Solution: As k becomes larger, the boundary becomes inflexible (linear). As k becomes smaller, the boundary becomes flexible (nonlinear). So in this case we would expect the best value for k to be large.

2) Assume that you are given an assignment which asks you to perform nested cross-validation with repeated resampling on a data set. Give the pseudo-code of your solution.

Solution:

- Determine the number of repetitions of sampling (with different seeds). Do resampling without replacement and create the data. For each sampled data:
  - Create r number of folds, use one fold for testing and the rest for training.
    - For each hyperparameter in the hyperparameter set (e.g. set of ks for k-NN), apply k-fold CV on the training set and obtain the performance.
    - Obtain the best hyperparameter based on the performances on $|k|$ validation sets.
  - Get the performance of the best model on the test set.
  - Obtain the performances over r-folds. Obtain the best model with hyperparameter based on voting.

3) Derive the lasso regularization for a linear regression of a scalar y on a p-dimensional vector x from the Bayesian point of view. Recall that

$$Laplace(\mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

Solution: Let $y = x^T\beta + \varepsilon$ denote the linear regression of y on x. Assume that we estimate β using a training sample of N observations (y_i, x_i), $i = 1, 2, \ldots, N$. Let $\mathbf{y}^T = (y_1, y_2, \ldots, y_N)$ and $\mathbf{X}^T = (x_1, x_2, \ldots, x_N)$. The posterior probability is calculated as

$$P(\beta|\mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}|\mathbf{X}, \beta)P(\beta)}{P(\mathbf{y}|\mathbf{X})}.$$

Our estimates of β are those values that maximize this posterior probability:

$$\begin{aligned}
\beta_{\text{MAP}} &= \arg\max_{\beta} P(\beta|\mathbf{y}, \mathbf{X}) && (1)\\
&= \arg\max_{\beta} \frac{P(\mathbf{y}|\mathbf{X}, \beta)P(\beta)}{P(\mathbf{y}|\mathbf{X})} && (2)\\
&= \arg\max_{\beta} P(\mathbf{y}|\mathbf{X}, \beta)P(\beta) && (3)\\
&= \arg\max_{\beta} \log(P(\mathbf{y}|\mathbf{X}, \beta)P(\beta)) && (4)\\
&= \arg\max_{\beta} \log(P(\mathbf{y}|\mathbf{X}, \beta)) + \log(P(\beta)) && (5)
\end{aligned}$$

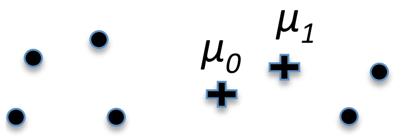
We can move from Equation (2) to Equation (3) since $P(\mathbf{y}|\mathbf{X})$ is constant. For lasso regression, we select our prior distribution as zero-mean Laplacean with variance b for each β_i. The likelihood function is given by

$$L(\beta|\mathbf{y}, \mathbf{X}) := P(\mathbf{y}|\mathbf{X}, \beta) = \prod_{i=1}^{N} P_Y(y_i|x_i, \beta, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\left(y_i - \left(\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p}\right)\right)^2}{2\sigma^2}}.$$

Inserting this prior and likelihood function we have:

$$\begin{aligned}
\beta_{\text{MAP}} &= \arg\max_{\beta} \left[\log \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\left(y_i - \left(\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p}\right)\right)^2}{2\sigma^2}} + \log \prod_{j=0}^{p} \frac{1}{2b} e^{-\frac{|\beta_j|}{b}} \right] && (6)\\
&= \arg\max_{\beta} \left[-\sum_{i=1}^{N} \frac{\left(y_i - (\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p})\right)^2}{2\sigma^2} - \sum_{j=0}^{p} \frac{|\beta_j|}{b} \right] && (7)\\
&= \arg\min_{\beta} \frac{1}{2\sigma^2} \left[\sum_{i=1}^{N} \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p})\right)^2 + \frac{2\sigma^2}{b} \sum_{j=0}^{p} |\beta_j| \right] && (8)\\
&= \arg\min_{\beta} \left[\sum_{i=1}^{N} \left(y_i - (\beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p})\right)^2 + \lambda \sum_{j=0}^{p} |\beta_j| \right] && (9)
\end{aligned}$$

4) Consider applying a Gaussian Mixture Model (GMM) with two components to cluster the data below. Let μ_0, μ_1, Σ_0 and Σ_1 denote the means and covariance matrices of these mixture components, and π_0 and π_1 to denote their mixture proportions (i.e., $p(x_i) = \pi_0 N(\mu_0, \Sigma_0) + \pi_1 N(\mu_1, \Sigma_1)$). We assume that the covariance matrices Σ_j in both components are diagonal. In the picture below, each dot represents a training observation. The '+' points indicate the current estimates of μ_0 and μ_1 of the two mixture components after the kth iteration of the EM-algorithm.



a. In which direction will the estimates of μ_0 and μ_1 move during the next M-step?

Solution: Assuming that the current estimates of the variances in the two cluster components are not too different, it is likely that the estimate of μ_0 will move to the left and the estimate of μ_1 will move to the right. For the first mixture component, the points to the left and right of μ_0 have about the same (posterior) probability. Given that their are four observations to the left and only two to the right, the estimate of μ_0 will move towards the larger 'cloud' of four observations. For the second mixture component, the two observations on the right will have a considerably higher posterior probability, such that the estimate of μ_1 will be drawn in their direction.

b. Will the marginal likelihood of the data increase or decrease on the next EM iteration? Explain your reasoning in one sentence.

Solution: Increase. Each iteration of the EM algorithm increases to likelihood of the data, unless you happen to be exactly at a local optimum.

c. Will the estimate of π_0 increase or decrease on the next EM step? Explain your reasoning.

Solution: Increase. π_0 is determined by adding the probabilities of all points that they are in cluster 1. In the current configuration, μ_1 is pretty close to μ_0, so that it will be 'stealing' a lot of this probability mass. As μ_0 moves to the left and μ_1 to the right, the probabilities to be in cluster 1 will increase substantially for the four observations on the left. While this probability will decline for the two observations on the right, overall we might expect the estimate of π_0 to increase.

5) Show that weighted Euclidean distance

$$d_e^{(w)}(x_{il}, x_{i'l}) = \frac{\sum_{l=1}^{p} w_l (x_{il} - x_{i'l})^2}{\sum_{k=1}^{p} w_k}$$

satisfies

$$d_e^{(w)}(x_i, x_{i'}) = d_e^{(}z_i, z_{i'}) = \sum_{l=1}^{p} (z_{il} - z_{i'l})^2$$

where

$$z_{il} = x_{il} \left(\frac{w_l}{\sum_{k=1}^{p} w_k} \right)^{1/2}.$$

Thus, weighted Euclidean distance based on x is equivalent to unweighted Euclidean distance based on z.

Solution:

$$\frac{\sum_{l=1}^{p} w_l (x_{il} - x_{i'l})^2}{\sum_{k=1}^{p} w_k} = \sum_{l=1}^{p} \left(\frac{w_l}{\sum_{k=1}^{p} w_k} \right) (x_{il} - x_{i'l})^2$$

Let $s_l = \dfrac{w_l}{\sum_{k=1}^{p} w_k}$

$$\sum_{l=1}^{p} s_l (x_{il} - x_{i'l})^2 = \sum_{l=1}^{p} (\sqrt{s_l} x_{il} - \sqrt{s_l} x_{i'l})^2$$

If we define vectors z_i with components given by $z_{il} = x_{il}\sqrt{s_l} = x_{il} \left(\frac{w_l}{\sum_{k=1}^{p} w_k} \right)^{1/2}$, it implies that

$$d_e^{(w)}(x_i, x_{i'}) = d_e(z_i, z_{i'})$$

What does this mean? Though the distances between data points are determined with the weighted distance, we can transform it into unweighted form and employ algorithms that use unweighted distances, e.g. K-means.

6) Suppose we train a single hidden layer feedforward neural network for a multinomial classification problem with K classes, using a training sample of N observations. Let x_i denote the p-dimensional input vector, and $y_{ik} = 1$ if observation i belongs to class k, and $y_{ik} = 0$ otherwise (with $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, K$). Assume that we use the softmax function to transform the outputs from the hidden layer, that is,

$$\hat{y}_{ik} = \frac{e^{-a_{ik}}}{\sum_{l=1}^{K} e^{-a_{il}}}$$

where $a_{ik} = \beta_{0k} + \beta_k^T Z_i$ with $Z_i = (Z_{i1}, Z_{i2}, \ldots, Z_{iM})$ and $Z_{im} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ for some activation function σ. Suppose we use the cross-entropy to train the neural network, that is, we estimate the vector of parameters θ by minimizing

$$E(\theta) = \sum_{i=1}^{N} E_i(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \ln \hat{y}_{ik}.$$

Show that $\partial E_i(\theta)/\partial a_{ij} = \hat{y}_{ij} - y_{ij}$.

Solution: Starting from $E_i(\theta) = -\sum_{k=1}^{K} y_{ik} \ln \hat{y}_{ik}$ with $\hat{y}_{ik} = e^{-a_{ik}} / \sum_{l=1}^{K} e^{-a_{il}}$, we find

$$\begin{aligned}
\frac{\partial E_i(\theta)}{\partial a_{ij}} &= -\sum_{k=1}^{K} y_{ik} \frac{1}{\hat{y}_{ik}} \frac{\partial \hat{y}_{ik}}{a_{ij}} \\
&= -y_{ij}(1 - \hat{y}_{ij}) - \sum_{k \neq j} y_{ik} \frac{1}{\hat{y}_{ik}} (\hat{y}_{ik} \hat{y}_{ij}) \\
&= -y_{ij}(1 - \hat{y}_{ij}) + \sum_{k \neq j} y_{ik} \hat{y}_{ij} \\
&= \sum_{k=1}^{K} y_{ik} \hat{y}_{ij} - y_{ij} \\
&= \hat{y}_{ij} - y_{ij}
\end{aligned}$$

7) Suppose that you are given the data set in Table 1, where x_1 and x_2 denote input variables and 'class' denotes the label of the output/target variable. Answer the following questions about a classification tree.

	x_1	x_2	class
S_1	black	0	1
S_2	black	1	1
S_3	red	0	2
S_4	black	1	1
S_5	red	0	1
S_6	red	1	2
S_7	red	0	2
S_8	black	1	1
S_9	red	1	1
S_{10}	red	0	1

Table 1: Dataset

a. Specify the splits that are possible in your tree.

Solution 1: Splits on x_1: black/red.

Splits on x_2: 0/1.

b. Calculate explicitly the gain for each split of the full training sample according to the Gini index. Which split should be selected?

Solution: The impurity in node m based on the Gini index is given by $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where K is the number of classes (equal to 2 in this case). The overall impurity is then equal to the weighted average of the Gini values for all nodes that results from a certain split.

$$
\begin{aligned}
I(x_1 = \text{red}) &= \frac{6}{10}\left[\left(\frac{3}{6} \times \frac{3}{6}\right) + \left(\frac{3}{6} \times \frac{3}{6}\right)\right] + \frac{4}{10}\left[(1 \times 0) + (0 \times 1)\right] \\
&= 0.30 \\
I(x_2 = 0) &= \frac{5}{10}\left[\left(\frac{3}{5} \times \frac{2}{5}\right) + \left(\frac{2}{5} \times \frac{3}{5}\right)\right] + \frac{5}{10}\left[\left(\frac{4}{5} \times \frac{1}{5}\right) + \left(\frac{1}{5} \times \frac{4}{5}\right)\right] \\
&= 0.40
\end{aligned}
$$

We thus select $I(x_1 = \text{red})$ as a split since it has the minimum Gini index.

c. Calculate explicitly the gain for each split of the full training sample according to Entropy. Which split should be selected?

Solution: The entropy in node m is given by $-\sum_{k=1}^{K}\hat{p}_{mk}(\log\hat{p}_{mk})$. The overall impurity is then equal to the weighted average of the Entropy values for all nodes that results from a certain split.

$$\begin{aligned}
I(x_1=\text{red}) &= P_1 I(2) + P_2 I(2)\\
&= \frac{6}{10}\left(-\frac{3}{6}\log\frac{3}{6}-\frac{3}{6}\log\frac{3}{6}\right)+\frac{4}{10}\left(-\frac{4}{4}\log\frac{4}{4}-\frac{0}{4}\log\frac{0}{4}\right)\\
&= 0.42\\
I(x_2=0) &= P_1 I(1) + P_2 I(2)\\
&= \frac{5}{10}\left(-\frac{3}{5}\log\frac{3}{5}-\frac{2}{5}\log\frac{2}{5}\right)+\frac{5}{10}\left(-\frac{4}{5}\log\frac{4}{5}-\frac{1}{5}\log\frac{1}{5}\right)\\
&= 0.59
\end{aligned}$$

We thus select $I(x_1 = \text{red})$ as a split since it has the minimum entropy.