CHAPTER

12

Data mining and machine learning techniques

12.1 Introduction

Although traditional safety data (e.g., crash data, highway, traffic, and environmental characteristics) remain the primary data sources for highway safety analysis, new and emerging data are becoming more and more available. The advent of autonomous and connected vehicle technologies and naturalistic driving studies offer far richer information than that of traditional data sources. Given large and complex datasets, we need to use advanced methods in addition to the strategies introduced in the preceding chapters to handle high-dimensional and nonlinear relationships in Big Data. In recent years, more people are turning their attention to data mining and machine learning techniques in hopes of discovering new, accurate and useful patterns through a "data-driven safety analysis." The readers should note that both data mining and machine learning fall under the umbrella of artificial intelligence (AI).

Data mining is the process of discovering and extracting useful information from a vast amount of data. Machine learning incorporates the principles and techniques of data mining and uses the same algorithms to automatically learn from and adapt to the data. The canonical characteristics in machine learning include, but are not limited to:

- *Clustering*: Clustering is used to create clusters where similar data points are grouped together. Applications in highway safety analysis include crash pattern recognition and hot spot identification. Typical methods include k-means and latent class clustering.
- *Classification*: Classification assigns data to one of several categories. Applications of classification in highway safety analysis include

https://doi.org/10.1016/B978-0-12-816818-9.00016-0

real-time crash prediction, crash injury severity prediction, crash types, and risky driver behaviors. Typical methods for such tasks include logistic regression, support vector machines, neural networks, random forests, and Bayesian network classifiers.
- *Regression*: Regression predicts continuous quantities from input data. Applications in highway safety analysis include prediction of crash counts, rates, and certain types of crash events. Typical methods include linear regression, neural networks, and Gaussian processes.

Examples of these three types of characteristics can be found in this chapter. Given the multitude of choices, it is important to consider how the data mining and machine learning techniques are used, what limitations exist, and whether there are explicit trade-offs (e.g., prediction and interpretation).

This chapter introduces data mining and machine learning methodologies and techniques that have been applied in highway safety studies, including association rules, clustering analysis, decision tree models, Bayesian networks, neural networks, and support vector machines. The theoretical frameworks are illustrated through exemplary cases published in safety literature and are supplemented with implementation information in the statistical software package R. The sensitivity analysis section at the end of the chapter offers a means of specifying the effect of an independent variable on the output, which can assist in deciding on the appropriate safety solutions. Machine learning is transforming the way we collect and analyze safety data; this chapter helps facilitate curriculum changes that improve data literacy and better prepare the future workforce for this transformation.

12.2 Association rules

Association rule is a rule-based machine learning method for discovering relations between variables in large databases. Specifically, it identifies the relative frequency of sets of variables (e.g., highway geometric features, traffic conditions, driver characteristics) occurring alone and together in a given event such as a crash. The rules have the form $A \rightarrow B$ in which A is the antecedent and B is the consequent. In association rules, the rules can be expressed by *Support*, *Confidence*, and *Lift*. The three indexes can be calculated as follows:

$$Support\ (A \rightarrow B) = \frac{\#(A \cap B)}{N}$$

$$Confidence = \frac{Support\ (A \rightarrow B)}{Support\ (A)} \tag{12.1}$$

$$Lift = \frac{Support\ (A \rightarrow B)}{Support\ (A) \times Support\ (B)}$$

where N is the number of crashes and # $(A \cap B)$ is the number of crashes in which both Conditions A (antecedent) and B (consequent) are presented.

As can be seen in Eq. (12.1), Support is the percentage of a rule that exists in the whole dataset. Confidence is the proportion of consequents among antecedents. Lift is used to quantify the statistical dependence because this rule indicates the number of cooccurrences of the antecedent and consequent to the expected cooccurrences, under the assumption that the antecedent and the consequent are independent. A Lift value smaller than 1 indicates a negative dependence between the antecedent and the consequent. A Lift value equal to 1 indicates independence, and a Lift value greater than 1 indicates positive dependence. For example, in the rule "reckless driving → alcohol (support = 1%, confidence = 50%, lift = 5)," Support shows that the proportion of observations including both reckless driving and alcohol is 1% in all crashes; Confidence shows that the proportion of observations including both reckless driving and alcohol is 50% in the dataset including reckless driving crashes only, or for 50% of the crashes involving reckless driving, the rule is correct (i.e., 50% of the times a reckless driving crash also involves alcohol); and Lift indicates that reckless driving is positively associated with alcohol.

The following four steps help create association rules: (1) generate all two-item rules; (2) determine threshold values; (3) eliminate the rules with lift values outside of the thresholds; and (4) eliminate the remaining rules that have both Support and Confidence values lower than the thresholds. In an effort to explore the types of driver errors leading to a crash at an intersection (Wang and Qin, 2015), the threshold value for Support is set to be 1%. For Confidence, the threshold is set to be 30% for both careless driving and reckless driving at uncontrolled and signal-controlled intersections. For stop-controlled intersections, the threshold for Confidence is set to be 10% for careless driving and 50% for reckless driving. For Lift, the threshold value is set to be greater than or equal to 1.1 (positive correlation), or smaller than or equal to 0.9 (negative correlation). The purpose of choosing different confidence thresholds is to accommodate different distributions of careless driving and reckless driving among the three types of intersections. As an example, Table 12.1 shows the rules that involve more severe mistakes (i.e., careless and reckless driving) for stop-controlled intersections.

TABLE 12.1 Association rules for driver errors for stop-controlled intersections.

	Careless driving			Reckless driving		
Antecedent	**Support**	**Confidence**	**Lift**	**Support**	**Confidence**	**Lift**
Old male	–	–	–	9%	87%	1.3
Young female	–	–	–	11%	87%	1.3
Middle-aged female	3%	11%	0.8	–	–	–
Old female	–	–	–	8%	87%	1.3
DUI	–	–	–	2%	80%	1.2
Horizontal curve	–	–	–	5%	54%	0.8
Vertical curve	2%	9%	0.7	10%	54%	0.8
Speed limit (35–55mph)	1%	34%	2.6	–	–	–
Speed limit (>55 mph)	1%	11%	0.8	3%	80%	1.2
Afternoon peak (4:00 p.m.–6:59 p.m.)	3%	10%	0.8	–	–	–
Nighttime (7:00 p.m.–6:59 a.m.)	5%	20%	1.5	–	–	–
Wet pavement	2%	10%	0.8	13%	74%	1.1
Snowy pavement	1%	10%	0.8	6%	54%	0.8
Icy pavement	–	–	–	2%	27%	0.4
Passenger car	–	–	–	76%	87%	1.3
Light truck	2%	22%	1.7	5%	54%	0.8
Heaver truck	1%	18%	1.4	–	–	–

Notes: "–" represents the rule is not applicable.
Source: Wang, K., Qin, X., 2015. Exploring driver error at intersections: key contributors and solutions. Transport. Res. Rec. 2514 (1), 1–9.

According to Table 12.1, the highest Support value for rules leading to careless driving is nighttime (5%). The highest Support value for rules leading to reckless driving is passenger car drivers (76%). Regarding Confidence, the highest value for rules leading to careless driving is a posted speed limit of 35–55 mph (34%). All Confidence values for rules leading to reckless driving are high except for poor pavement conditions. High Confidence values are due to the high percentage of reckless driving (67%) taking place at sign-controlled intersections.

12.3 Clustering analysis

In data mining, clustering analysis (CA) is an unsupervised learning technique with a principal objective of dividing a dataset into smaller subsets called clusters. CA is based on a heuristic method and tries to maximize the similarity between intracluster elements and the dissimilarity between intercluster elements (Fraley and Raftery, 2002). The two clustering methods most commonly used in safety research are K-means clustering (KC) and latent class clustering (LCC).

12.3.1 K-means clustering

KC is a nonhierarchical, similarity-based clustering method that partitions the observations in K clusters, based on the distance of the observations from clusters' means. Many algorithms can be used to partition a dataset, including naïve k-means (or Lloyd's algorithm), Forgy and Random Partition, and the Hartigan–Wong method. Before running the KC algorithm, a distance function and the value of K need to be specified. A popular choice for the distance function is the Euclidean distance. The value of K can be determined visually through a Scree plot that exhibits different K values versus the corresponding results in terms of the intracluster homogeneity. An example of the K-means algorithm is presented in detail in Section 10.4 of Chapter 10—*Capacity, Mobility and Safety*, Characterizing Crashes by Real-Time Traffic.

12.3.2 Latent class cluster

LCC is a model-based clustering method that assumes data from a mixture of probability densities. Given $Y_1, \ldots, Y_n$, each described by a set of features $(y_1, \ldots, y_m)$, the mixture probability density for the whole dataset can be expressed as follows (Hagenaars and McCutcheon, 2002):

$$f(\boldsymbol{y_i}|\theta) = \sum_{k=1}^{K} \pi(C_k) f_k(\boldsymbol{y_i}|C_k, \theta_k) \tag{12.2}$$

where $\boldsymbol{y_i}$ denotes the ith object's scores on a set of observed variables, K is the number of clusters, π_k denotes the prior probability of belonging to the latent class k (or membership), and θ are the class-specific parameters. So, the distribution of $\boldsymbol{y_i}$ given the model parameters θ, $f(\boldsymbol{y_i}|\theta)$, is a mixture of class-specific densities $f_k(\boldsymbol{y_i}|C_k, \theta_k)$. Maximum-likelihood (ML) and maximum-posterior (MAP) are the two main estimation methods for LCC models, and most software packages use an expectation-maximization (EM) algorithm to find ML or MAP estimates.

Like the KC model, the number of clusters can be visually determined through the Scree plot. A better way to evaluate and decide the number of clusters in an LCC model is to use information criteria such as AIC, BIC, and CAIC (see Section 2.7 of Chapter 2—*Fundamentals and Data Collection*).

The model with a lower score of BIC, AIC, or CAIC is considered to be better. Thus, LCC employs statistical criteria to decide the most appropriate number of clusters and allows probability classifications to be determined by using subsequent membership probabilities estimated with the maximum likelihood method.

To further assess the quality of the clustering solution, the entropy R^2 can be calculated (McLachlan and Peel, 2000) as in Eq. (12.3) where p_{ik} denotes the posterior probability that case i belongs to cluster k.

$$Entropy\ R^2 = 1 - \frac{-\sum_{i=1}^{n}\sum_{k=1}^{K} p_{ik}\ln(p_{ik})}{n\ln(K)} \tag{12.3}$$

In a perfect case of classification, the criterion equals to 1; a worst-case scenario for clustering would have a value of 0 for the criterion.

In a study by Depaire et al. (2008), 29 crash-contributing factors (e.g., crash, vehicle, driver, environmental) were used to partition 4028 crashes. The results of cluster analysis in Table 12.2 indicate distinctions

TABLE 12.2 Feature probabilities by cluster.

Variable − value	C1 (%)	C2 (%)	C3 (%)	C4 (%)	C5 (%)	C6 (%)	C7 (%)
Accident type: collision with a pedestrian	0	99	0	0	0	99	6
Crossroad: crossroad without traffic lights or priority road	74	26	40	21	40	15	5
Crossroad: no crossroad	5	48	1	56	33	76	42
Built-up area: outside built-up area	1	1	1	1	1	0	47
Road type: highway, national, regional or provincial road	25	25	55	23	24	7	99
Age road user 1: 0−18 years old	14	11	16	16	42	96	11
Dynamics road user 2: road user is not moving	0	11	0	79	74	35	0
Vehicle type road user 1: motorcycle or bicycle	8	0	12	17	90	0	7
Vehicle type road user 1: car	85	0	81	76	9	0	82
Size (%)	23	19	15	15	14	10	4

Source: Depaire, B., Wets, G., Vanhoof, K., 2008. Traffic accident segmentation by means of latent class clustering. Accid. Anal. Prev. 40 (4), 1257−1266.

between traffic crashes with cars, motorcycles, or bicycles and whether pedestrians were involved. The analysis shows that other types of features, such as type of crossroad or road type, can be used to group data. The seven-cluster model provides cluster-specific distributions for each variable, which helps to characterize every cluster as a unique and specific type.

12.4 Decision tree model

Among the various data mining techniques, outcomes from tree-based models are relatively simple for nonstatisticians to interpret. A tree-based classification and regression model can be constructed by recursively partitioning data with criteria such as the total sum of the squared error (SSE). In other words, the values of all the variables in the model, either discrete or continuous, are selected to yield the maximum reduction in the variability of the response variable. The algorithm obtains optimal results by exhaustively searching all variables as well as all values for each selected variable.

12.4.1 The CART model

The classification and regression trees (CART) methodology, first introduced by Breiman et al. (1998), refers to two types of decision trees: a classification tree and a regression tree. A classification tree is an algorithm where the response variable is categorical, such as crash injury severity levels. The algorithm is used to identify the "class" to which a response variable would most likely belong. A regression tree refers to an algorithm where the response variable is a continuous variable, such as crash frequency or crash rate, and the algorithm is used to predict its value.

The CART model has been used to predict crash injury severity, crash frequency, and to identify important contributing factors based on the tree structure. The decision tree grows when the dataset is split into groups and organized so that each group's data is as homogenous as possible. Algorithms for constructing decision trees usually work top-down by choosing a splitter at each step that best splits that particular set of variables. Although different algorithms use different metrics, they usually measure the homogeneity of the target variable within the subsets using one of the following:

- *Gini impurity*: Gini impurity is a measure of how often an element from the dataset would be labeled wrong if it is randomly labeled according to the distribution of labels in the subset

$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$. Here, the Gini impurity of a node n is 1 minus the sum of all J classes of the fraction in each class (p_i) squared.

- *Variance reduction*: The variance reduction of a node is defined as the total reduction of the variance of the target variable due to the split at the node. It is often used when the target variable is continuous, such as in a regression tree.
- *Information gain*: The information gain is based on the concept of entropy and information content. At each step, the split chosen is usually the one that results in the purest children nodes. Information gain is used to decide which feature to split at each step when building the tree. It is used by the ID3, C4.5, and C5.0 decision-tree algorithms.

The notations used in the tree algorithm are as follows: t denotes a node, T denotes a tree, $\widetilde{T}$ denotes the set of terminal nodes of T, $\left|\widetilde{T}\right|$ denotes the number of terminal nodes of T, T_t denotes a subtree of T with root node t, and $\{t\}$ denotes a subtree of T_t containing only the root node t. The following describes the theory of a Poisson regression and classification using the tree algorithm notations.

- First, crash data are modeled by the Poisson distribution when the equality of the mean and variance is not violated, or by the Negative Binomial distribution when data overdispersion is present. If the number of crashes, the response variable Y_i, is assumed to follow a Poisson distribution, the expected number of crashes, μ_i, can be expressed as the product of traffic exposure and the exponential function of the potential crash contributing factors or other explanatory variables, as follows: $\mu_i = V^{*}exp(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik})$. The linear relationship between the expected number of crashes and the corresponding vector of predictors is obtained from the logarithm transformation and the maximum likelihood estimation of $\beta_0, \beta_1, \ldots, \beta_k$ can be obtained by maximizing the likelihood function

$L(\boldsymbol{\mu}|\boldsymbol{y}) = \prod_{i=1}^{n} e^{-\mu_i} \mu_i^{y_i} \Big/ y_i!$ or equivalently, the log-likelihood function $LL(\boldsymbol{\mu}|\boldsymbol{y}) = \sum_{i=1}^{n} y \ln(\mu_i) - \sum_{i=1}^{n} \mu_i - \sum_{i=1}^{n} y_i!$.

- Then, when partitioning a dataset recursively, an appropriate splitter needs to be selected. If the predictor variable x_i is numerically ordered, the dataset is partitioned by $x_i \le c$, and if x_i is categorical, the dataset is partitioned by $x_i \in A$, where c is a constant

and A is a fixed subset of possible values of x_i. The method proceeds by an iterative search for the variable as well as its specific value from all of the variables within all the possible levels or values in the model that result in the maximum reduction in variability of the dependent variable. The best splitter, s^*, is determined by deviance D or by squared errors, where a squared error for a node t is defined as follows:

$$D(t) = \sum_{x \in I} (y_n - \hat{\mu})^2 \tag{12.4}$$

where $\hat{\mu}$ is an estimation of the mean or a sample mean $\overline{y}$. For generalized linear models, the deviance is also called the log-likelihood (ratio) statistic, which is described in Section 2.7 of Chapter 2—*Fundamentals and Data Collection*. In the Poisson case, deviance can be simplified as

$$D = 2 \times \left\{ \sum_{i=1}^{n} y_i \ln(y_i/\hat{\mu}) - \sum_{i=1}^{n} (y_i - \hat{\mu}) \right\}$$

If the deviance for a node t is denoted by $D(t)$, the deviance for a tree T is

$$D(T) = \sum_{t \in \widetilde{T}} D(t) = \sum_{t \in \widetilde{T}} \sum_{x \in t} (y_n - \overline{y}(t))^2 \tag{12.5}$$

For a binary partitioning by a splitter s, a difference by s is defined as

$$\Delta D(s,t) = D(t) - D(t_L) - D(t_R) \tag{12.6}$$

where t_L and t_R are the left and right child nodes of t, respectively.

- Finally, the best splitter, s^*, is obtained by maximizing the difference, that is,

$$\Delta D(s^*,t) = \max_{s \in S} \Delta D(s,t) \tag{12.7}$$

where S is the set of all possible splitters.

The maximum reduction occurs at a specific value of a selected variable s. When the data are split at s into two samples, these remaining samples have a much smaller variance in Y than the original dataset. Thus, the reduction at node t is the greatest when the deviances at nodes t_L and t_R are smallest.

In a study by Qin and Han (2008), classification criteria have been developed to categorize sites into groups sharing similar attributes and consequences. A tree-based regression method significantly improves the

model efficiency. The variables used for the CART model include total number of crashes (dependent variable), type of area (rural or urban), types of traffic controls (all-way, side, signal), number of intersection approach legs (3, 4, or other), number of major roadway lanes (2, 4, or unknown), existence of major roadway median, and existence of left-turn lane(s). In Fig. 12.1, the CART tree shows that the first splitter to predict the crash rate at an intersection is the number of intersection approaches, followed by intersection traffic control types.

R 3.5.0 (R Core Team, 2018) includes two available packages, "Rpart" and "Tree," to help build classification and regression trees. The key difference between the two packages is the way missing values are handled during the splitting and scoring processes. In "Tree," an observation with a missing value for the primary split rule is terminated. "Rpart" offers more flexibility, allowing users to decide how to handle missing values by using surrogates to set up the "usesurrogate" parameter in the rpart.control option.

Despite its convenience, the CART model has several limitations. First, the CART model cannot quantitatively measure the effect of variables on injury severity as there is not an estimated coefficient for each variable. Second, the CART model predicts the outcome based on a single decision tree whose classification accuracy can be unstable due to the data, split variable, and complexity of tree change (Chung, 2013). Third, the CART model may cause an overfitting problem (Duncan and Khattak, 1998).

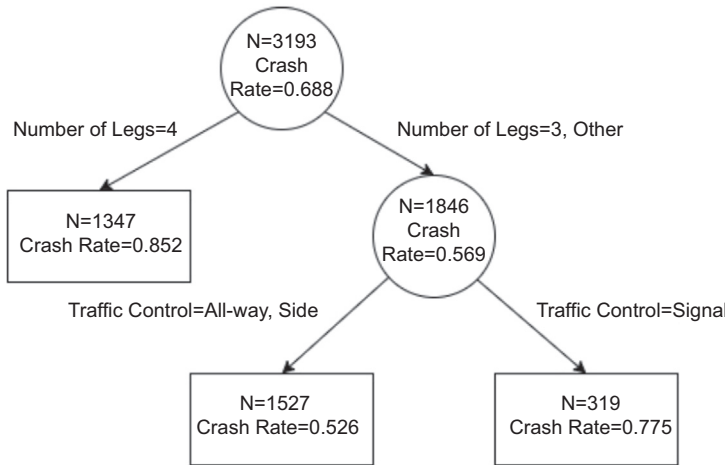

FIGURE 12.1 Cart model results.

An alternative to the CART model is ensemble learning methods, which can be used to predict responses by growing a series of classification trees. The classification trees are grown by randomly selected samples with replacement (i.e., bootstrap samples). Random forest (RF) and gradient boosting trees (GBT) are representative methods for ensemble learning methods. The RF classifier is a type of bootstrap aggregation (i.e., bagged) decision tree in which the ensemble method builds multiple decision trees by repeatedly resampling training data with replacement data. The trees then vote for a consensus prediction. GBT, on the other hand, builds trees one at a time, and each new tree helps correct mistakes made by the previously trained tree.

12.4.2 Random forest

The RF method generates many classifiers and aggregates their results. Breiman (2001) proposed this method as a prediction tool that uses a collection of tree-structured classifiers with independent, and identically distributed random vectors. Each tree votes for the most popular class. The method performs very well compared with many other classifiers and is robust against the overfitting problem, one of CART's limitations. The procedures for implementing RF are summarized in Algorithm 12.1 (Lee and Li, 2015).

Algorithm 12.1: Random forest algorithm

1. Select a bootstrap sample.
2. Grow a classification tree to fit the bootstrap sample so that the variable can be selected only from a small subset of randomly selected variables for each split in the classification tree.
3. Predict the response variable for the samples not selected in the bootstrap sample (i.e., out-of-bag samples) by using the classification tree in Step 2. The predicted category of the response variable is the category with the highest proportion of samples.
4. Compare the observed and predicted categories of the response variable to calculate the rate of incorrect classification of the sample (the number of misclassified samples over the total number of samples) for each tree. This rate is defined as the misclassification rate (r_b).

continued

Algorithm 12.1: Random forest algorithm (*cont'd*)

5. For each predictor variable i, permute the value of the variable in the out-of-bag samples. Predict the response variable by using the classification tree in Step 2 to calculate the new misclassification rate of the tree (r_{ai}). The importance score for variable i is computed on the basis of the difference between the misclassification rates before and after the permutation $[(r_{ai} - r_b)/r_b]$. A higher difference between the two misclassification rates increases the importance score, meaning the variable importance is higher. The importance score for each variable is updated as more trees are trained to the out-of-bag samples.
6. Repeat Steps 1–5 until enough trees are grown by using different bootstrap samples. Calculate the average importance score for each variable in different trees.

RF uses a random sample of the data to train each tree independently. This randomness helps to make the model more robust than a single tree and less likely to overfit the training data. However, RF does not identify whether a variable has a positive or negative effect on the response variable. Hence, RFs are often used to rank the importance of variables as a screening method for selecting input variables for other models such as a logistic regression. For a categorical variable with multiple levels, RFs are biased in favor of the attribute values with more observations and may produce unreliable variable importance scores. In addition, a large quantity of trees may make the algorithm slow for real-time prediction. The RF technique is implemented in the R package "randomForest" that is based on Breiman's (2001) method.

12.4.3 Gradient boosted trees

Gradient boosting is a machine learning technique for regression and classification problems. Friedman (2001) introduced the technique as the gradient boosting machine. The technique typically uses decision trees (especially CART trees) of a fixed size as base learners, so it is often called as GBT. Unlike RF, which builds an ensemble of deep independent trees, GBT builds an ensemble of shallow and weak trees sequentially. Each new tree in GBT learns and improves on the previously trained tree by applying a higher weight to incorrectly classified observations and a lower weight to correctly classified observations. The chance that higher weights will be correctly classified increases when the weak learners are boosted. Hence, the GBT model transforms an ensemble of weak

learners into a single strong model and predicts the cases that are difficult to classify.

In the GBT model, a basis function $f(x)$ describes a response variable y in a function of the summation of weighted basis functions for individual trees as follows:

$$f(x) = \sum_{n=1}^{m} \beta_n b(x; \gamma_n) \tag{12.8}$$

where $b(x; \gamma_n)$ is the basis function for individual tree n; m is the total number of trees; γ_n is the split variable; and, β_n is the estimated parameter that minimizes the loss function, $L(y, f(x))$. GBT can be summarized in Algorithm 12.2 (Friedman, 2001):

Algorithm 12.2: GBT algorithm

1. Initialize $f_0(x)$, which can be set to zero.
2. For $n = 1,2,3, \ldots, m$ (number of trees)
   a. For $i = 1$ to k (number of observations), calculate the residual r.
      $r = -\frac{\partial L(y, f(x))}{\partial f(x)}$ where $L(y, f(x)) = (y - f(x))^2$; $f(x) = f_{m-1}(x)$ and
      $f_{m-1}(x)$ is the basis function for the previous tree $(m-1)$.
   b. Fit a decision tree to r to estimate γ_n
   c. Estimate β_n by minimizing $L(y_i, f_{n-1}(x)) + \beta_n b(x; \gamma_n)$
   d. Update $f_n(x) = f_{n-1}(x) + \beta_n b(x; \gamma_n)$
3. Calculate $f(x) = \sum_{n=1}^{m} \beta_n b(x; \gamma_n)$

The GBT model can handle different types of predictor variables and can also accommodate missing data. The GBT model can fit complex nonlinear relationships and automatically account for interactions between predictors. As boosted trees are built by optimizing an objective function, GBT can be used to solve almost all objective functions as long as the gradient functions are available. In addition, boosting focuses step by step on difficult cases is an effective strategy for handling unbalanced datasets because it strengthens the impact of positive cases. However, GBT training generally takes time because trees are built sequentially, and each tree is built to be shallow. Therefore, the quantitative effect of each variable on the response variable, such as crash injury severity, may not be available. A study by Lee and Li (2015) compares the model performances of CART and GBT. The R package implements the "Generalized Boosting Model" method in "gbm." Interested readers can refer to Elith and Leathwick (2017) for details on building GBT with "gbm."

12.5 Bayesian networks

Bayesian network (BN) is a probabilistic graphical model that depicts a set of variables and their conditional dependencies via a directed acyclic graph (DAG). BN has two main components: the causal network model (topology) and the conditional probability tables (CPTs). The model causal relationships are represented as DAGs in which variables are denoted by nodes, and relationships (e.g., causality, relevance) among variables are described by arcs between nodes. CPTs explicitly specify the dependencies among variables in terms of conditional probability distributions.

Let $U = \{x_1, \ldots, x_n\}$, $n \geq 1$ be a set of variables and B_p be a set of CPTs, $B_p = \{p(x_i \mid p_a(x_i), x_i \in U)\}$ where $p_a(x_i)$ is the set of parents of x_i in BN and $i = (1, 2, 3, \ldots, n)$. A BN represents joint probability distributions P(U):

$$P(U) = \prod_{x_i \in U_p} P(x_i | P_a(x_i)) \tag{12.9}$$

Bayes' theorem can be applied to predict any variable in U given the other variables using $p(x_i|x_j) = \frac{p(x_j|x_i)p(x_i)}{p(x_j)}$. For instance, the classification task consists of classifying a variable y given a set of attribute variables U. A classifier h: $U \rightarrow y$ is a function that maps an instance of U to a value of y. The classifier is learned from a dataset consisting of samples over (U, y).

The arcs in the BN model could represent causality, relevance, or relations of direct dependence between variables. In highway safety analysis, it is better to consider arcs as evidence of a direct dependence between the linked variables rather than view them as causality due to the complexity of crashes. The absence of arcs means the absence of direct dependence between variables, which does not necessarily mean the absence of indirect dependence between variables.

Candidate BNs are assigned a goodness-of-fit "network score" to be maximized by heuristic algorithms. Classes of heuristic algorithms include greedy hill climbing (HC), genetic algorithms, tabu search and simulated annealing. The R package includes "bnlearn" to help explain the graphical structure of BN, estimate its parameters and perform useful inferences. Functions in "bnlearn" include HC and tabu, as well as severity score functions such as AIC and BIC.

Prati et al. (2017) investigated factors related to the severity of bicycle crashes in Italy using the Bayesian network analysis. The DAG in Fig. 12.2 shows the association between the severity of bicycle crashes and crash characteristics. The network consists of nine nodes, one for the target and one for each predictor. The BN model also indicates the relative importance of each predictor, using a darker color for more important

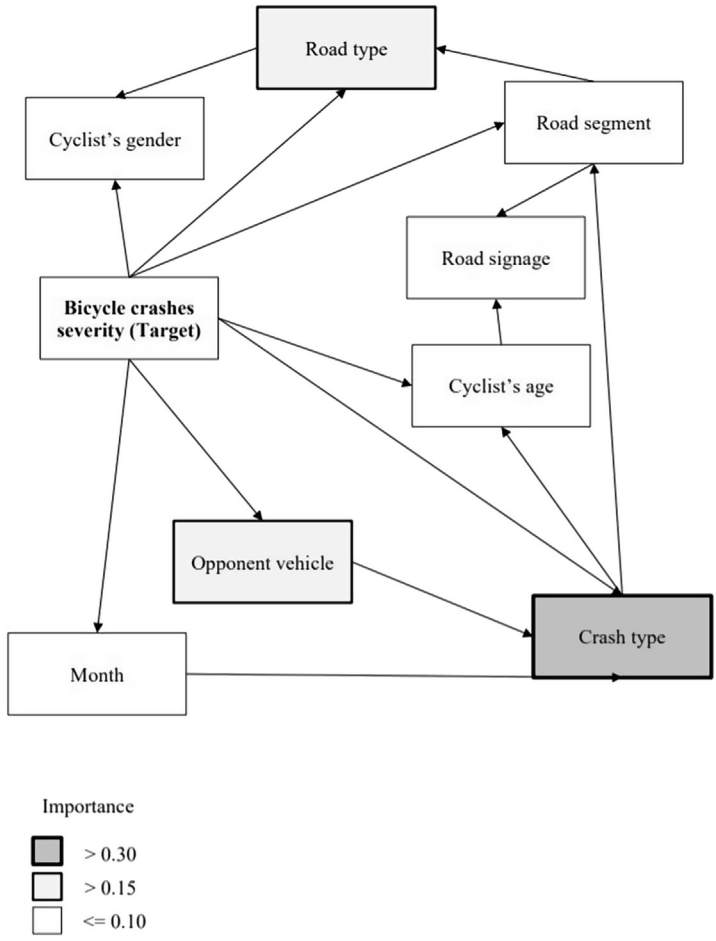

FIGURE 12.2 The Bayesian network model and predictor importance (Prati et al., 2017).

relationships to the severity of bicycle crashes: crash type (0.31), road type (0.19), and type of opponent vehicle (0.18).

The BN model provides a conditional probability table for each node in which each column represents a value of the predictor and each row represents a combination of values of the target and parent predictor variables. Table 12.3 summarizes the conditional probability for each crash type by its parents (i.e., a combination of bicycle crash severity and month). The conditional probabilities of crash type suggest that in angle crashes, fatalities are less likely to occur than injuries, especially from February to December. Fatalities were more likely than injuries in rear-end crashes.

TABLE 12.3 Crash type/month conditional probabilities (Prati et al., 2017).

Month	Severity of bicycle crashes	Run-off-the-road	Head-on collision	Sudden braking	Angle collision	Falling from the vehicle	Sideswipe collision	Rear-end collision	Hit pedestrian	Hit parked or stationary vehicle	Hit stopped vehicle	Hit obstacle in carriageway
January	Fatality	0.06	0.09	0	0.51	0.05	0.13	0.13	0	0	0	0.02
February	Fatality	0.03	0.08	0	0.43	0.05	0.08	0.32	0	0.02	0	0
March	Fatality	0.08	0.11	0	0.41	0.09	0.09	0.19	0	0.03	0.01	0
April	Fatality	0.05	0.03	0	0.31	0.09	0.15	0.34	0	0	0.01	0
May	Fatality	0.03	0.15	0	0.4	0.06	0.08	0.19	0	0.02	0	0.06
June	Fatality	0.04	0.11	0	0.35	0.04	0.12	0.31	0	0.03	0	0.01
July	Fatality	0.05	0.15	0	0.35	0.07	0.15	0.24	0	0	0	0
August	Fatality	0.03	0.05	0	0.41	0.03	0.21	0.27	0	0.01	0	0
September	Fatality	0.05	0.15	0	0.42	0.04	0.14	0.19	0.01	0	0	0
October	Fatality	0.07	0.05	0	0.31	0.06	0.2	0.2	0.01	0.05	0	0.03
November	Fatality	0.01	0.11	0.02	0.31	0.06	0.21	0.26	0	0.02	0	0
December	Fatality	0	0.07	0	0.42	0.05	0.15	0.28	0	0.03	0	0
January	Injury	0.03	0.08	0	0.53	0.03	0.18	0.07	0	0.05	0.01	0.01
February	Injury	0.03	0.07	0	0.54	0.04	0.18	0.07	0	0.05	0.01	0.01
March	Injury	0.05	0.06	0	0.51	0.03	0.19	0.08	0	0.05	0.01	0.02
April	Injury	0.04	0.06	0	0.51	0.04	0.2	0.07	0.01	0.05	0.01	0.01

May	Injury	0.04	0.06	0	0.5	0.04	0.21	0.07	0.01	0.05	0.01	0.02
June	Injury	0.04	0.07	0	0.49	0.04	0.2	0.08	0.01	0.05	0	0.02
July	Injury	0.04	0.07	0	0.5	0.03	0.2	0.08	0	0.06	0.01	0.02
August	Injury	0.04	0.07	0	0.5	0.04	0.2	0.08	0.01	0.05	0	0.02
September	Injury	0.04	0.06	0	0.49	0.03	0.22	0.08	0	0.05	0.01	0.01
October	Injury	0.04	0.05	0	0.53	0.04	0.19	0.07	0.01	0.06	0.01	0.01
November	Injury	0.03	0.07	0	0.52	0.02	0.19	0.08	0	0.07	0.01	0.01
December	Injury	0.03	0.05	0	0.57	0.03	0.17	0.08	0	0.06	0	0.01

12.6 Neural network

The artificial neural network (ANN) is a machine learning technique used to model the response in a large dataset as a nonlinear (activation) function of linearly combined predictors. ANN is a means of effectively discovering new patterns and correctly classifying data or making forecasts. The output signal from one neuron can be used as an input for other neurons, thus effectively modeling and solving complex problems through a network of multiple neurons and multiple layers. Several types of ANNs are described in this section.

12.6.1 Multilayer perceptron neural network

The feed-forward neural network (FNN) effectively solves multivariate nonlinear regression and classification problems. The multilayer perceptron (MLP) neural network is a class of FNN in which neurons of the same layer are not connected to each other, but connected to the neurons of the preceding and subsequent layers. An output of one hidden layer serves as an input to the subsequent layer in the form of the activation function of a weighted summation of the outputs of the last hidden layer. The weights can be determined by solving the optimization problem or minimizing a given cost function. The algorithm most commonly used to determine the weights is back-propagation.

In Fig. 12.3, the MLP model, Xs are the independent variables (indicators), Hs are the hidden nodes, and Y is the dependent variable. αs are the estimated coefficients between hidden nodes and indicators, and βs are the estimated coefficients between hidden nodes and dependent variables. The model can be described as follows:

$$Y = g_Y(\beta_0 + \beta_1 H_1 + \ldots + \beta_{m-1} H_{m-1}) + \boldsymbol{\varepsilon} \tag{12.10}$$

where $H_i = g_H\left(\sum_j \alpha_{i,j} X_j\right)$ $j = 0,\ 1,\ \ldots p-1$; ε is the error term; and g_Y and g_H are the activation functions. In ANN, the activation function of a node defines the output of that node given an input. Fig. 12.4 shows common activation functions such as the radial basis function (e.g., Gaussian), the sigmoidal function (e.g., logistic), the hyperbolic function (e.g., tanh), the rectified linear unit (ReLU), and the identity function. For example, if the activation function is the logistic function as $g(z) = (1 + e^{-z})^{-1}$, then the model can be transformed to

$$\begin{aligned} Y &= \left[1 + \exp\left[-\beta_0 - \sum\nolimits_{n=1}^{m-1} \beta_n \left[1 + \exp\left(-\alpha_{n,j} X_j\right)\right]^{-1}\right]\right]^{-1} + \boldsymbol{\varepsilon} \\ &= f(\boldsymbol{X},\ \boldsymbol{\alpha},\ \boldsymbol{\beta}) + \boldsymbol{\varepsilon} \end{aligned} \tag{12.11}$$

Data normalization can improve data fitting and model performance given different types and various magnitudes of input variables;

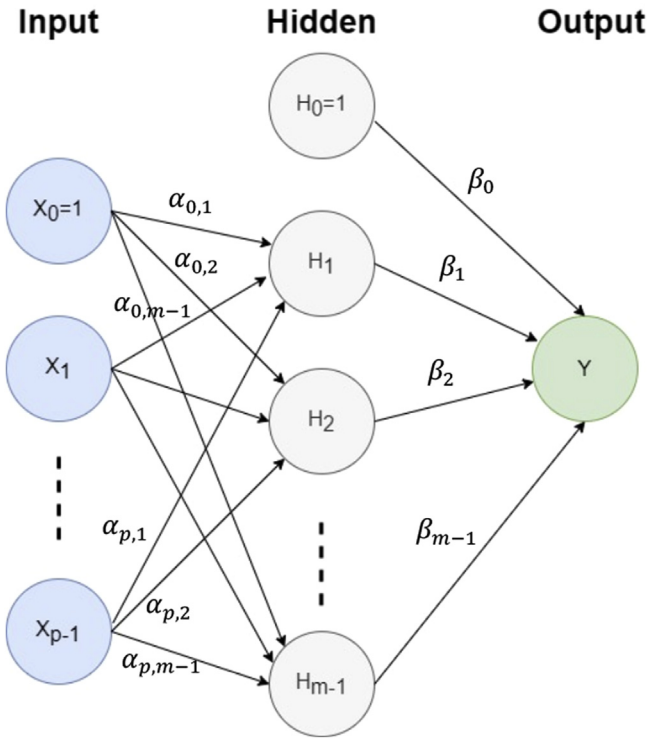

FIGURE 12.3 The multilayer perceptron (MLP) ANN model.

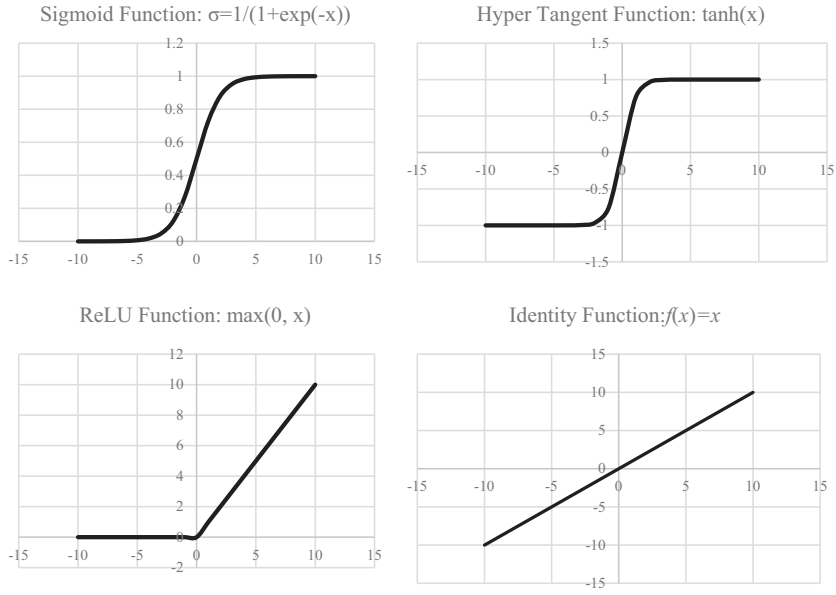

FIGURE 12.4 Common activation functions.

therefore, data normalization is required for ANN. Variables are normalized using the min-max normalization formula where the minimum of 0 and maximum of 1 are used to match the lower and upper limits of the sigmoid activation function in ANN models. Categorical variables with more than two values are converted into (N-1) binary variables before the normalization. The normalization equation is as follows:

$$x_{ni} = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{12.12}$$

Statistical software R has several packages for dealing with ANNs, such as "neuralnet," "nnet," and "RSNNS." However, not all packages include the ability to plot the function that allows analysts to visualize the neural networks. Additionally, the ability to take separate or combined x and y inputs as data frames or as a formula may also not be included. Software package development is an evolving process, so readers are encouraged to check regularly for new developments and compare the differences between these options in R.

ANNs are not constrained by a predetermined functional form or specific distributional assumptions, so they are expected to produce more accurate functions that predict the number of crashes given the explanatory variables. Kononov et al. (2011) applied ANNs to explore the underlying relationships between crashes and other factors for urban freeway segments. The crash-frequency models (or SPFs in this case) developed from a sigmoid functional form through ANN with one hidden layer showed a better model fit when compared with a traditional NB regression model.

Several studies have used different types of ANN models, including MLP, to predict driver injury severity or identify significant factors of injury severity in traffic crashes in hopes of better understanding the relationships between driver injury severity and factors related to the driver, vehicle, roadway, and environment (Abdelwahab and Abdel-Aty, 2002; Delen et al., 2006). The modeling results in these studies, when compared to traditional methods such as MNL and the ordered logit model, are promising. However, ANN models, like other classification models, have the multiclass classification problem, meaning they ignore less-represented categories to improve the model's overall accuracy. This is a problem for injury severity studies as the data are highly skewed due to the presence of fewer high-severe injury crashes and more less-severe injury crashes. One solution is reducing the multiclass problem into a series of two-class (binary) classification problems. It can be argued that multiple binary models are more advantageous than a single multiclass model in that they provide better insight into the interrelationships between different levels of injury severity and the crash factors (Delen et al., 2006; Li et al., 2012).

Resampling is another technique used to handle the multiclass classification problem. Resampling involves oversampling less-representative

classes, undersampling overly-representative classes, or using ensemble methods (e.g., Bootstrap aggregating). Jeong et al. (2018) used undersampling, oversampling, and ensemble methods (majority voting and bagging) to classify motor vehicle injury severity. Yuan et al. (2019) used the Synthetic Minority Over-sampling Technique (SMOTE) algorithm for unbalanced classification problems to predict real-time crash risk in their long short-term memory recurrent neural networks. Interested readers can refer to Chawla et al. (2002) for details on SMOTE. Long short-term memory recurrent neural networks will be described in more detail below.

For specific data types, such as image or time-series, MLP may not be a good choice of method. MLP does not scale well for image data because it uses one perceptron for each input, and the number of weights can grow rapidly for large images. Additionally, MLP ignores information based on the position of a pixel and its spatial correlation with neighbors, resulting in the loss of spatial information when the image is flattened into an MLP. For time-series data, MLP ignores the time sequence between data points. Hence, for image or time-series data, other neural network algorithms such as convolutional neural networks and recurrent neural networks should be considered.

12.6.2 Convolutional neural networks

Convolutional neural networks (CNNs, or ConvNet) are deep learning neural networks that are commonly applied to image analyses. The name "convolutional" indicates that the algorithm employs a mathematical operation called "convolution." CNN typically consists of convolutional layers, pooling layers, and fully connected layers. The convolutional layer is the core building block of CNN.

As shown in Fig. 12.5, a convolutional layer creates a filter that slides over the image spatially, resulting in a feature map. Various filters can produce many separate feature maps that are stacked to generate volume.

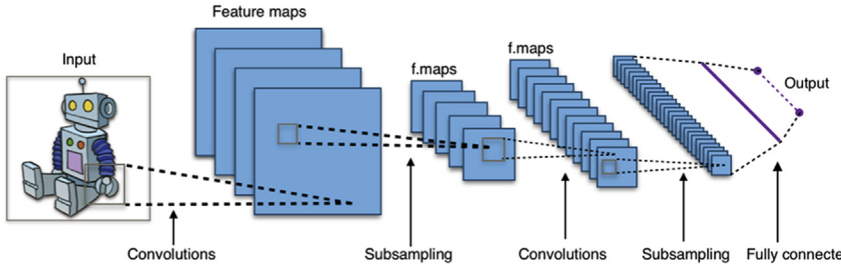

FIGURE 12.5 Convolutional neural networks (CNN) structure.

The results are then passed to the next layer. Pooling the layers reduces the size of the convoluted feature by combining the outputs of node clusters at one layer into a single node in the next layer. Pooling is a form of nonlinear down-sampling where max pooling is the most common among several nonlinear functions. Pooling increases computational efficiency through dimensionality reduction. Finally, after several convolutional and max-pooling layers, the important features of an image can be understood. The matrix that represents the extracted features will be flattened and fed into a traditional MLP neutral network for classification purposes. Image recognition in the statistical software R 3.5.0 (R Core Team, 2018) uses deep CNN in the "MXNet" package.

Ma et al. (2017) applied the CNN to analyze time series freeway speed data for incident detection. First, freeway speed time series data were converted to images by the Gramian Difference Angular Field (GDAP) method. Then, CNNs were used to identify high-level features in the image and predict the probability of a crash. The authors used the AlexNet structure (Krizhevsky et al., 2012) that contains eight layers, including five convolutional layers and three fully connected layers. In the fully connected layer, the ReLU activation function was used to convert the two-dimensional image to a one-dimensional vector and understand complex feature interactions. Finally, the sigmoid layer was used to predict the probability of a crash. The study used 5000 samples, which included 212 crash events and 4788 noncrash events. The dataset was split into training and testing sets. The model performance was evaluated by the detection rate and the false alarm rate, and the AUC of the CNN model is equal to 0.9662.

12.6.3 Long short-term memory—recurrent neural networks

Recurrent neural networks (RNN) are a very important variant of neural networks that are heavily used in natural language processing. In RNN, connections between nodes form a directed graph following a temporal sequence, allowing temporal patterns in the data to be modeled. RNNs can retain a "memory" that is captured in the time-series data.

Long short-term memory (LSTM) is a deep learning RNN. As shown in Fig. 12.6, a common LSTM unit includes a cell (C_t), an input gate (I_t), an output gate (O_t), and a forget gate (F_t). The cell remembers values over certain time intervals, and the three gates regulate the flow of information into and out of the cell. The cell trains the model by using back-propagation. RNN that uses LSTM units partially solves the vanishing gradient problem (i.e., the gradients can tend to zero), a phenomenon that occurs when training an RNN using back-propagation.

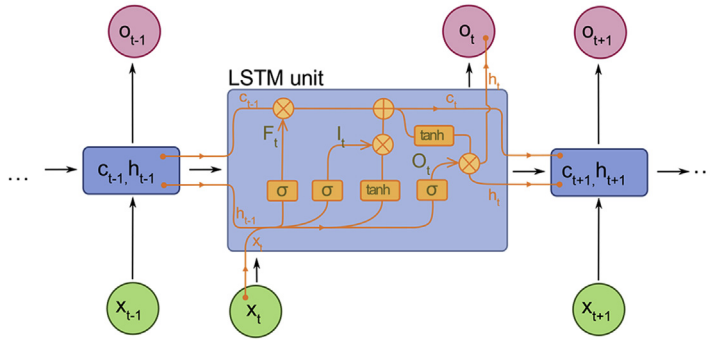

FIGURE 12.6 Long short-term memory (LSTM).

The input gate decides what values from the input should be used to modify the memory. For example, a sigmoid function decides what values to let through, and a tan*h* function assigns weights to the passing values for their levels of importance, ranging from −1 to 1. The forget gate determines what memories the cell can forget. For example, a sigmoid function outputs a number between 0 and 1 for each number in the cell state C_{t-1} based on the previous state (h_{t-1}) and the content input (X_t). The output gate yields the output based on the input and the memory of the cell. For example, functioning similar to the input gate, a sigmoid function decides what values to let through. A tan*h* function gives weights to the passing values for their levels of importance ranging from −1 to 1 and is then multiplied with the output of the sigmoid function.

LSTM with a forget gate can be formulated as follows (Hochreiter and Schmidhuber, 1997; Gers et al., 1999):

$$\begin{aligned} \boldsymbol{f_t} &= \sigma(\boldsymbol{W_f x_t} + \boldsymbol{U_f h_{t-1}} + \boldsymbol{b_f}) \\ \boldsymbol{i_t} &= \sigma(\boldsymbol{W_i x_t} + \boldsymbol{U_i h_{t-1}} + \boldsymbol{b_i}) \\ \boldsymbol{o_t} &= \sigma(\boldsymbol{W_o x_t} + \boldsymbol{U_o h_{t-1}} + \boldsymbol{b_o}) \\ \boldsymbol{c_t} &= i_t \circ \tanh(\boldsymbol{W_c x_t} + \boldsymbol{U_c h_{t-1}} + \boldsymbol{b_c}) + \boldsymbol{f_t} \circ \boldsymbol{c_{t-1}} \\ \boldsymbol{h_t} &= o_t \circ \tanh(\boldsymbol{c_t}) \\ \boldsymbol{y_t} &= \boldsymbol{W_y y_{t-1}} + \boldsymbol{b_y} \end{aligned} \tag{12.13}$$

where $\boldsymbol{x_t}$ is the input vector to the LSTM unit and $\boldsymbol{y_t}$ is the output vector; $\boldsymbol{f_t}$ is the forget gate's activation vector; $\boldsymbol{i_t}$ is the input/update gate's activation vector; $\boldsymbol{o_t}$ is the output gate's activation vector; $\boldsymbol{h_t}$ is the hidden state vector, also known as the output vector of the LSTM unit, and $\boldsymbol{c_t}$ is the cell state vector; $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{b}$ are weight matrices and bias vector

parameters, respectively. σ is the sigmoid function; and, tan$h$ is the hyperbolic tangent function. The term "$\circ$" denotes the Hadamard product or the element-wise product of two matrices. Initial values are $\boldsymbol{c}_0 = 0$, $\boldsymbol{h}_0 = 0$.

The R 3.5.0 (R Core Team, 2018) package "rnn" implements LSTM, gated recurrent unit (GRU), and vanilla RNN models. The "keras" R package, an open-source neural-network library written in Python, may be another option. "keras" was developed to enable fast experimentation and supports both convolution-based networks and recurrent networks. The R interface for "H_2O," is another choice. H_2O is a fully open-source machine learning platform that offers parallelized implementations of many supervised and unsupervised machine learning algorithms such as Generalized Linear Models, Gradient Boosting Machines (including XGBoost), Random Forests, Deep Neural Networks (Deep Learning), Stacked Ensembles, Naive Bayes, Cox Proportional Hazards, K-Means, PCA, Word2Vec, as well as a fully automatic machine learning algorithm (AutoML) (https://cran.r-project.org/web/packages/h2o/index.html).

In highway safety literature, Yuan et al. (2019) used an LSTM recurrent neural network to predict real-time crash risk. The authors attempted to predict 5-min crash risk that is updated every minute at signalized intersections. Crash data, travel speed, signal timing, loop detector, and weather data were collected from 44 signalized intersections in Oviedo, FL. The LSTM-RNN was applied, and the results were compared with a conditional logistic model based on a matched case-control design. The comparison results show that the LSTM-RNN with the synesthetic minority oversampling technique outperforms the conditional logistic model.

12.6.4 Bayesian neural networks

The Bayesian neuaral network (BNN) model was initially proposed by Liang (2003, 2005) where a fully connected MLP neural network structure with one hidden layer was applied in Fig. 12.3. For the BNN model, the transfer functions used in the hidden layer and the output layer are the same as those used in the MLP model.

Although the network structure of the proposed BNN model is very similar to the MLP structure, they are different in the prediction mechanism and the training process. An example is given to illustrate the differences in the prediction mechanism. Assume that there are n sets of crash data $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where the definitions of x_i and y_i are the same as what we used for the NB regression and MLP models. Let θ denote all the network parameters or weights, β_j, α_k, and γ_{jk}

($j = 1,\ldots,M$; $k = 0,\ldots,P$), in Fig. 12.3. The predicted number of crashes for site i using BNNs is given by Eq. (12.14).

$$\widehat{y}_i = \int f_B(x_i,\theta) \times P(\theta|(x_1,y_1),\ldots,(x_n,y_n))d\theta \tag{12.14}$$

where $f_B(x_i,\theta)$ is defined as

$$f_B(x_i,\theta) = \alpha_0 + \sum_{k=1}^{P}\left(\alpha_k * x_{ik}\right) + \sum_{j=1}^{M}\left\{\beta_j * \tanh\left(\sum_{k=1}^{P}\gamma_{jk} * x_{ik} + \gamma_{j0}\right)\right\} \tag{12.15}$$

$P(\theta|(x_1,y_1),\ldots,(x_n,y_n))$ in Eq. (12.14) is the posterior distribution of θ given observed data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$. One can see the main difference between BNNs and MLPs is that for BNNs the network parameter θ follows a certain probability distribution, and the prediction process for BNNs is to evaluate the integral of $f_B(x_i,\theta) \times P(\theta|(x_1,y_1),\ldots,(x_n,y_n))$ over all possible values of θ, while for MLPs the network parameter is fixed.

The actual BNN model is more complicated than the example given earlier. Readers are referred to Liang (2003, 2005) for a more detailed description of the BNN model and its Evolutionary Monte Carlo (EMC) training algorithm, as well as an application of using BNN to predict motor vehicle crashes (Xie et al., 2007).

12.7 Support vector machines

A support vector machine (SVM) is a machine learning approach originally developed by Cortes and Vapnik (1995), Vapnik (1998). SVM includes a set of supervised learning methods that can be used for classification and regression analysis.

A simple two-class classification problem is illustrated in Fig. 12.7. First, the input data points are mapped from data space to a high-dimensional feature space using a nonlinear kernel function such as a Gaussian kernel. The SVM model then constructs two separating hyperplanes (see dashed line in Fig. 12.7A) in the high dimensional space to separate the outcome into two classes so that the distance between them is as large as possible. The region bounded by the two separating hyperplanes is called the "margin," and the optimal separating hyperplane is in the middle (see the solid line in Fig. 12.7A). The idea is to search for the optimal separating hyperplane by maximizing the margin between the classes' closest points. The points lying on the boundaries are called support vectors. Fig. 12.7B represents a typical neural network with one input layer, one hidden layer, and one output layer.

Assume the training input is defined as vectors $x_i \in R^{\mathrm{In}}$ for i = 1, 2,…, N, which are the set of explanatory variables, and the training output is

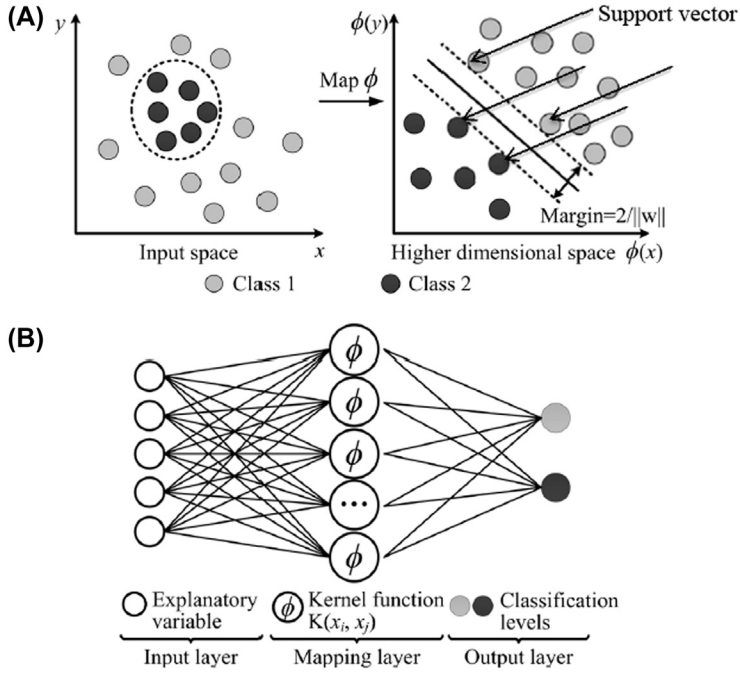

FIGURE 12.7 Classification of SVM models (Li et al., 2012).

defined as $y_i \in R^1$, which is the crash injury severity level. The SVM maps x_i into a feature space $R^h (h > \text{In})$ with higher dimension using a function $\Phi(x_i)$ to linearize the nonlinear relationship between x_i and y_i. The estimation function of y_i is

$$\widehat{y} = w^T \varphi(x) + b \tag{12.16}$$

where w is a normal vector that is perpendicular to the hyperplane. Both w and b are coefficients that are derived by solving the following optimization problem.

For the two-class classification problem, given a training set of instance-label pairs $(x_i,\ x_j)$, the SVM model aims to solve the optimization problem in Eq. (12.17) (Cortes and Vapnik, 1995).

$$\min_{w,\ b,\ \xi} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \tag{12.17}$$

Subject to

$$y_i \left(w^T \varphi(x_i) + b \right) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$

where ξ_i is a slack variable that measures misclassification error; C is a regularization parameter that is the penalty factor to errors (e.g., a large parameter C value indicates a small margin and vice versa). The coefficient C is still undetermined, but this optimization problem can be solved using the Lagrange multiplier:

$$\max\min\left\{\frac{1}{2}w^T w + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left[y_i\left(w^T\varphi(x_i)+b\right)-1+\xi_i\right]-\sum_{i=1}^{N}\beta_i\xi_i\right\} \tag{12.18}$$

where $\alpha_i,\ \beta_i > 0$ are Lagrange multipliers. The max sign means that among all hyperplanes separating the data, one is unique in yielding the maximum margin of separation between the classes.

$\varphi\left(x_i^T\right)\varphi(x_j)$ is the kernel function where a radial basis function (RBF) is often used and is defined as follows:

$$\varphi\left(x_i^T\right)\varphi(x_j) = K(x_i,\ x_j) = \exp\left\{-\gamma\|x_i - x_j\|^2\right\} \tag{12.19}$$

where γ is a parameter. Sometimes γ is parameterized as $\frac{1}{2\sigma^2}$. $\|x_i - x_j\|^2$ and may be recognized as the squared Euclidean distance between the two feature vectors. This model can be easily extended for handling multiclass classification tasks such as crash injury severity with the KABCO scale.

An alternative is called ν-SVC, where the parameter C is replaced by a parameter $\nu \in [0, 1]$. ν is used to control the number of support vectors. An additional variable ρ needs to be optimized to remove the user-chosen error penalty factor C. Introducing a new variable ρ adds another degree of freedom to the margin (Scholkopf et al., 2000). The optimization problem is formulated as follows:

$$\min_{w,\xi,\rho,b}\ \frac{1}{2}w^T w - \nu\rho + \frac{1}{N}\sum_{i=1}^{N}\xi_i \tag{12.20}$$

Subject to

$$y_i\left(w^T\varphi(x_i)+b\right) \geq \rho - \xi_i,\ \ \forall i = 1, \cdots,\ N$$

$$\xi_i \geq 0 \quad \forall i = 1, \cdots,\ N$$

$$\rho \geq 0$$

The R interface to "libsvm" is in package "e1071" where svm(·) includes C- classification, ν-classification, one-class-classification (novelty detection), ε-regression and ν-regression, svm() also includes linear, polynomial, radial basis function, and sigmoidal kernels, formula interface, and k-fold cross-validation. For further implementation details on libsvm, see Chang and Lin (2001).

SVM's superior performance has led to its popularity in highway safety analysis, for injury severity classification and for crash count prediction. Li et al. (2012) applied the C-SVM models to predict injury severity of crashes at freeway diverge areas. The SVM model was better at predicting crash injury severity when compared with the ordered probit (OP) model. However, the performance of the SVM model depends highly on the learning procedure, which contains functional mapping and parameter selection. The authors suggested using kernel functions other than the basic RBF kernel to improve the model performance. Li et al. (2008) applied v-SVM to predict motor vehicle crashes on frontage roads in Texas. The SVM model results were compared with traditional NB regression models, and several sample sizes were evaluated for the examination of data fitting and model prediction capabilities. The SVM models were consistently lower regarding the values of mean absolute deviation and mean absolute percentage error, for all sample sizes than those for NB regression models.

12.8 Sensitivity analysis

Despite their excellent performance in prediction, machine learning techniques such as ANN and SVM have been long criticized for performing as a black-box solution that cannot be directly used to identify the relationship between outcomes and input variables. To safety professionals and researchers, identifying crash contributing factors and quantifying their effects in terms of direction and magnitude is critical to informed decision-making, targeted investment, and improved safety. One of the methods safety practitioners have used to address this concern is the sensitivity analysis (Delen et al., 2006; Li et al., 2008, 2012).

The sensitivity analysis studies how the uncertainty in the output of a mathematical model can be attributed to different sources of uncertainty in its input. The sensitivity analysis can be used to measure the relationship between the input variables and the output of a trained neural network model (Principe et al., 2000). In the process of performing a sensitivity analysis, the neural network learning ability is disabled so that the network weights are not affected. Each input variable of a black-box model (e.g., ANN, SVM) is perturbed by a user-defined amount, with the other variables being fixed at their respective means or medians. The results before and after the perturbation of each input variable are recorded, and the impacts of each input variable on the output are calculated. For example, in the crash injury severity level prediction, the percent change of each severity level by one unit change of an input variable can be estimated.

References

Abdelwahab, H.T., Abdel-Aty, M.A., 2002. Artificial neural networks and logit models for traffic safety analysis of toll plazas. Transport. Res. Rec. J. Transport. Res. Board 1784, 115–125.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (No. 1), 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1998. Classification and Regression Trees. Chapman & Hall/CRC, Boca Raton, FL.

Chang, C.-C., Lin, C.-J., 2001. Training v-support vector classifiers: theory and algorithms. Neural Comput. 13 (9), 2119–2147.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.

Chung, Y.S., 2013. Factor complexity of crash occurrence: An empirical demonstration using boosted regression trees. Accident Anal. Prev. 61, 107–118.

Cortes, C., Vapnik, V., 1995. Support-vector network. Mach. Learn. 20, 273–297.

Delen, D., Sharda, R., Bessonov, M., 2006. Identifying significant predictors of injury severity in traffic accidents using a series of artificial neural networks. Accid. Anal. Prev. 38 (3), 434–444.

Depaire, B., Wets, G., Vanhoof, K., 2008. Traffic accident segmentation by means of latent class clustering. Accid. Anal. Prev. 40 (4), 1257–1266.

Duncan, C.S., Khattak, A.J., 1998. Applying the ordered probit model to injury severity in truck–passenger car rear-end collisions. In: Transportation Research Record 1635. TRB, National Research Council, Washington, D.C, pp. 63–71.

Elith, J., Leathwick, J., 2017. Boosted Regression Trees for Ecological Modeling. https://cran.r-project.org/web/packages/dismo/vignettes/brt.pdf.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232.

Fraley, C., Raftery, A.E., 2002. Model-based clustering, discriminant analysis, and density estimation. J. Am. Stat. Assoc. 97 (458), 611–631.

Gers, F.A., Schmidhuber, J., Cummins, F., 1999. Learning to Forget: Continual Prediction with LSTM.

Hagenaars, J.A., McCutcheon, A.L. (Eds.), 2002. Applied Latent Class Analysis. Cambridge University Press.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Jeong, H., Jang, Y., Bowman, P.J., Masoud, N., 2018. Classification of motor vehicle crash injury severity: a hybrid approach for imbalanced data. Accid. Anal. Prev. 120, 250–261.

Kononov, J., Lyon, C., Allery, B., 2011. Relation of flow, speed, and density of urban freeways to functional form of a safety performance function. Transport. Res. Rec. J. Transport. Res. Board 2236, 11–19.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 1097–1105.

Lee, C., Li, X., 2015. Predicting driver injury severity in single-vehicle and two-vehicle crashes with boosted regression trees. Transport. Res. Rec. 2514 (1), 138–148.

Li, X., Lord, D., Zhang, Y., Xie, Y., 2008. Predicting motor vehicle crashes using support vector machine models. Accid. Anal. Prev. 40 (4), 1611–1618.

Li, Z., Liu, P., Wang, W., Xu, C., 2012. Using support vector machine models for crash injury severity analysis. Accid. Anal. Prev. 45, 478–486.

Liang, F., 2003. An effective Bayesian neural network classifier with a comparison study to support vector machine. Neural Comput. 15 (8), 1959–1989.

Liang, F., 2005. Bayesian neural networks for nonlinear time series forecasting. Stat. Comput. 15 (1), 13–29.

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., Wang, Y., 2017. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. Sensors 17 (4), 818.

McLachlan, G.J., Peel, D., 2000. Finite Mixture Models. Wiley, New York.

Prati, G., Pietrantoni, L., Fraboni, F., 2017. Using data mining techniques to predict the severity of bicycle crashes. Accid. Anal. Prev. 101, 44–54.

Principe, J.C., Euliano, N.R., Lefebre, W.C., 2000. Neural and Adaptive Systems: Fundamentals through Simulations. John Wiley and Sons, New York.

R Core Team, 2018. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Qin, X., Han, J., 2008. Variable selection issues in tree-based regression models. Transport. Res. Rec. 2061 (1), 30–38.

Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L., 2000. New support vector algorithms. Neural Comput. 12, 1207–1245.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.

Wang, K., Qin, X., 2015. Exploring driver error at intersections: key contributors and solutions. Transport. Res. Rec. 2514 (1), 1–9.

Xie, Y., Lord, D., Zhang, Y., 2007. Predicting motor vehicle collisions using Bayesian neural networks: an empirical analysis. Accid. Anal. Prev. 39 (5), 922–933.

Yuan, J., Abdel-Aty, M., Gong, Y., Cai, Q., 2019. Real-time crash risk prediction using long short-term memory recurrent neural network. Trans. Res. Rec. 2673 (4), 314–326.