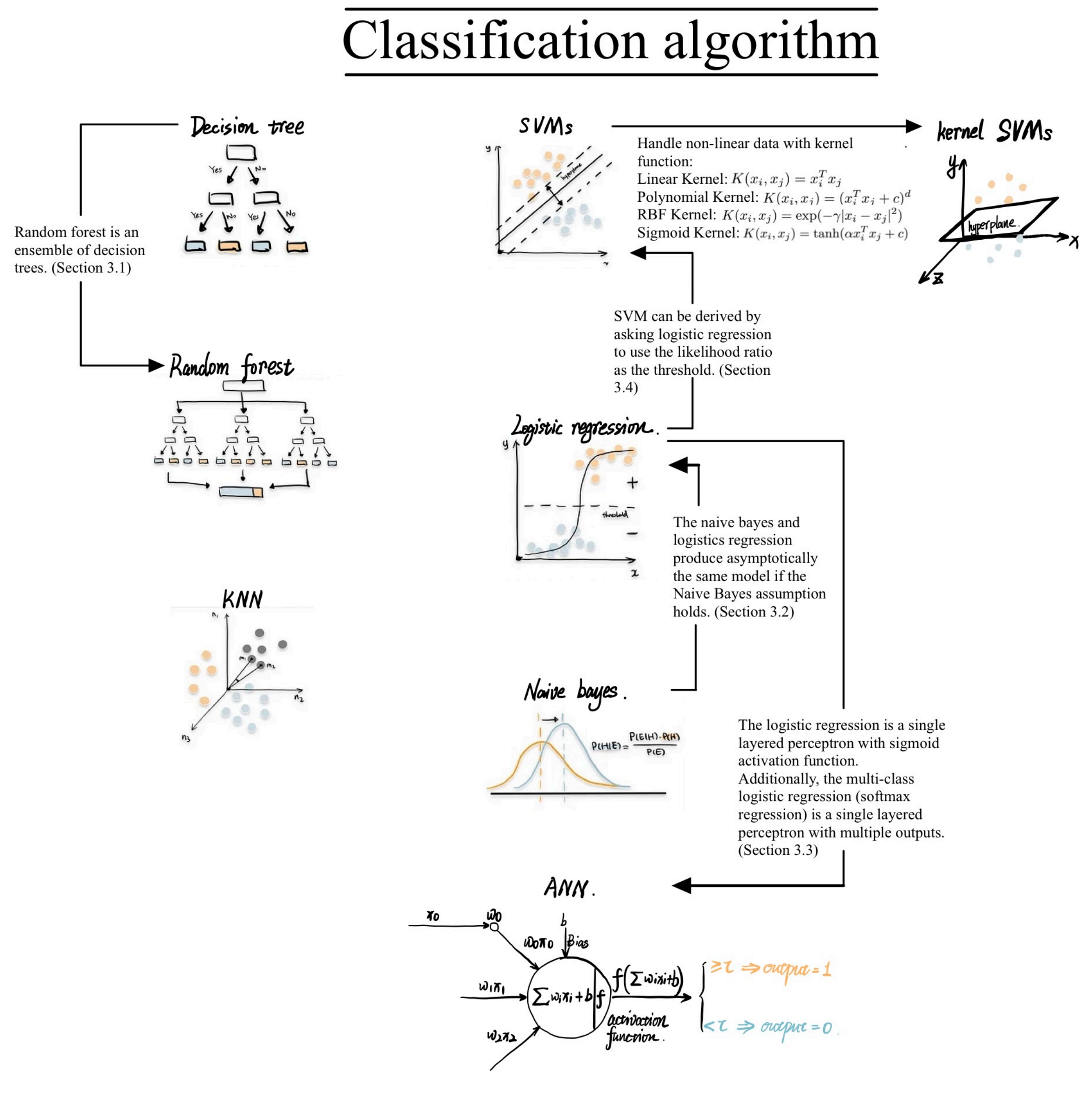
AMA546 Statistical Data Mining
Tutorial 6 Summary of classification algorithms

Contents

1 Introduction

The classification algorithm is a type of the supervised learning algorithms where the instances in a dataset have known labels or corresponding correct outputs. The goal is to learn a function that maps the inputs to the correct discrete labels.



Random forest is an ensemble of decision trees. (Section 3.1)

Handle non-linear data with kernel function:
Linear Kernel: $K(x_i, x_j) = x_i^T x_j$
Polynomial Kernel: $K(x_i, x_i) = (x_i^T x_i + c)^d$
RBF Kernel: $K(x_i, x_j) = \exp(-\gamma |x_i - x_j|^2)$
Sigmoid Kernel: $K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$

SVM can be derived by asking logistic regression to use the likelihood ratio as the threshold. (Section 3.4)

The naive bayes and logistics regression produce asymptotically the same model if the Naive Bayes assumption holds. (Section 3.2)

The logistic regression is a single layered perceptron with sigmoid activation function. Additionally, the multi-class logistic regression (softmax regression) is a single layered perceptron with multiple outputs. (Section 3.3)

Figure 1: Relationship of classification algorithms

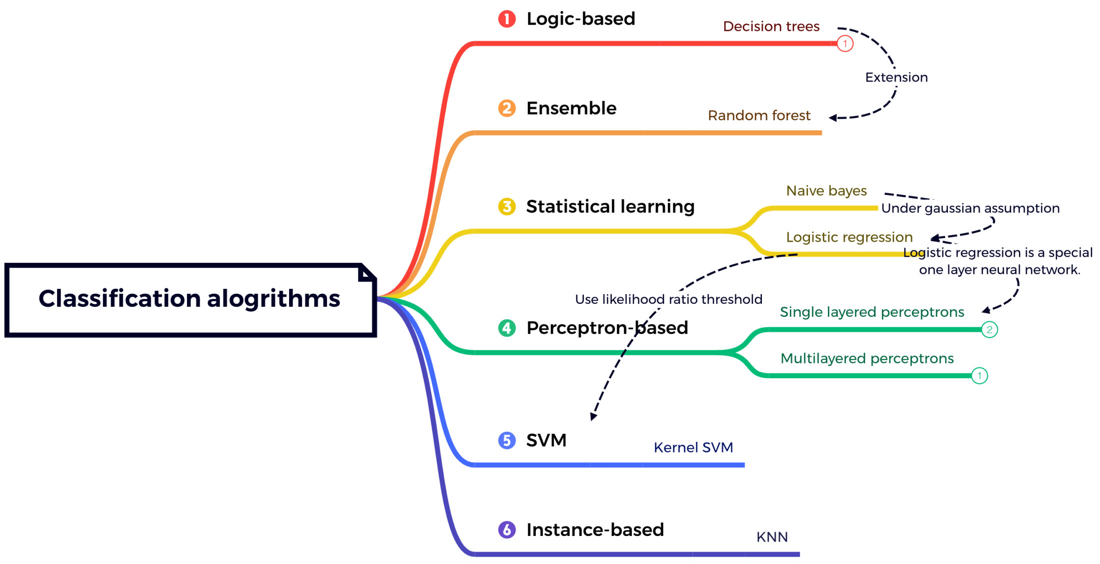


Figure 2: General structure

In this tutorial, we will covers the classification algorithms that we have already studied in this semester. As in fig1, except SVM, each algorithm are summarized by a topic. This means that each topic contains algorithms that are not covered in the course, and you can try to supplement them. The relationships between algorithms are connected by dashed lines, which we will cover these in detail below.

Since we have already learnt the detailed algorithms, the introduction of the classification algorithms will be very brief. We will focus on the relationship and classification of the algorithms in this tutorial. In section 2, we make a short review of the algorithms. Some relationships between the algorithms are discussed in section 3. The comparison of the algorithms are presented in section 4.

2 Classification algorithms

2.1 Logic-based algorithm: Decision tree

A logic-based algorithm is a type of algorithm that uses logical rules and reasoning to solve a problem or make a decision. The logical rules are typically represented in the form of if-then statements, which describe the conditions under which a particular action should be taken. It works by analyzing a set of inputs and then applying a series of logical rules to those inputs to arrive at a conclusion.

Decision tree classifier is a tree-like structure where internal nodes represent a test on an attribute, branches represent the outcomes of the test, and leaf nodes represent the class labels or continuous target values. Normally, the decision tree algorithm recursively selects the attribute that best separates the data based on some criterion (such as information gain or Gini index) and creates a binary split. The algorithm continues this process on each child node until it reaches a stopping criterion, such as a maximum depth or a minimum number of instances per leaf.

Decision tree has several advantages, such as being easy to understand and interpret, handling both categorical and continuous data, and being able to capture non-linear relationships between features. It can also handle missing values and outliers. However, decision tree is prone to overfitting, especially when the tree is deep and complex. This can be addressed by pruning or using ensemble methods like random forests.

2.2 Ensemble method: Random forest

Ensemble learning is combines multiple models to improve prediction accuracy and reduce overfitting. Random Forest is a popular ensemble method that builds multiple decision trees and combines their predictions.

Each decision tree in a Random Forest is built using a random subset of the features (Feature sampling) and a random subset of the data (Bootstrap aggregating). This helps to reduce the variance of each individual tree and improve the overall performance of the forest. During training, each tree is grown to its maximum depth is reached or no further improvement in the impurity measure can be achieved, and then the final prediction is made by combining the outputs of all the trees in the forest.

One of the advantages of Random Forest is its ability to handle missing data and noisy features. It is also computationally efficient and can handle large datasets with high-dimensional feature spaces.

2.3 Statistical learning algorithm

The statistical approaches are characterized by having an explicit underlying probability model, which provides a probability that an instance belongs in each class, rather than simply a classification.

2.3.1 Naive Bayes

Naive Bayes is based on Bayes' theorem, which states that the probability of a hypothesis given the evidence is proportional to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \propto P(E|H)P(H)$$

where:

- P(H|E) is the probability of hypothesis H occurring given the evidence E has occurred
- P(E|H) is the probability of evidence E occurring given that hypothesis H has occurred
- P(H) is the prior probability of hypothesis H
- P(E) is the prior probability of evidence E

Naive Bayes assumes that the features used to classify the data are independent of each other, which is called the "naive" assumption. The algorithm calculates the probability of each class given the input data, and then chooses the class with the highest probability as the output. This process involves calculating the probability of each feature given each class, as well as the prior probability of each class.

Naive Bayes is a fast and efficient algorithm, and it can work well even with small amounts of data. It is commonly used in text classification, spam filtering, and other applications where there are many features and the data is sparse. However, its assumption of feature independence can lead to inaccurate results in some cases.

2.3.2 Logistic regression

Logistic regression is a popular statistical method for binary classification tasks, where the goal is to predict the probability of an input belonging to one of two classes. The logistic regression assumes a linear relationship between the input features and the log odds of belonging to one of the classes. The logistic function $\sigma(z) = \frac{1}{1+e^{-z}}$ is then applied to this linear combination to map it to a probability value between 0 and 1.

The logistic regression model can be formulated mathematically as:

$$p(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p)}}$$

where x represents the input features, β_0 represents the intercept term, and $\beta_1, \ldots, \beta_p$ represent the coefficients of the input features. The logistic function maps the linear combination of input features and coefficients to a probability value, where $y = 1$ represents the positive class and $y = 0$ represents the negative class. The model is trained using a maximum likelihood estimation approach.

The Multi-class logistic regression, also known as softmax regression, is an extension of logistic regression for multi-class classification problems. In this case, instead of modeling the probability of a binary outcome, it models the probability of each class using a softmax function, which outputs a vector of probabilities that sum to 1. The softmax function is defined as:

$$p(y=j|\mathbf{x}) = \frac{e^{\beta_j^\top \mathbf{x}}}{\sum_{k=1}^{K} e^{\beta_k^\top \mathbf{x}}}$$

where j is the class index, $\mathbf{x}$ is the input feature vector, β_j is the weight vector for class j, and K is the total number of classes. Similar to logistic regression, the optimal values of β for multi-class logistic regression are found using maximum likelihood estimation and numerical optimization methods.

The logistic regression is a simple and interpretable model that can handle linearly separable data and is robust to noise. However, it may not perform well on datasets with nonlinear decision boundaries, where more complex models such as decision trees or neural networks may be more appropriate.

2.4 Perceptron-based algorithm: Single layered perceptron

Perceptron is a type of artificial neural network (ANN) used for binary classification problems. It borrows the idea from the neuron in the human brain. The single layered perceptron is a linear algorithm that learns a linear function to separate the classes. The perceptron receives input values, applies weights to them, sums them up, and passes the result through an activation function to produce a binary output:

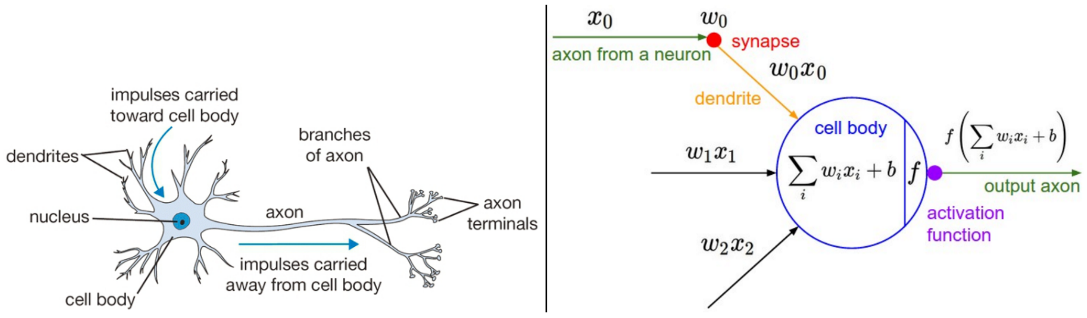


Figure 3: Human neuron and Single layered perceptron

In Figure 3: The left graph is the biological neuron: There are about 86 billion neurons in the human nervous system, and they are connected by about $10^14 - 10^15$ synapses. Each neuron receives input signals from its dendrites and produces output signals along its only axon, which gradually branches at the end. It's connected by synapses to the dendrites of other neurons.

The right graph is the mathematical model of neuron: The signal propagated by the axon (x_0) is based on the sensitivity of the axon to the signal (w_0). The strength of the synapse (weight vector (w_0)) can be learned and controls the connection strength to other neurons, which can be expressed as excitation or inhibition of the signal. In the cell body, All the weighted signals are added and compared to a threshold to obtain the output signal, usually Sigmoid function is used, which will map the summed signal to between (0,1).

Perceptron-based techniques are simple to implement and computationally efficient, making them suitable for large datasets. However, they may not perform well for complex classification tasks with nonlinear decision boundaries, where other techniques such as support vector machines may be more effective.

2.5 Support Vector Machine

2.5.1 SVM

Support Vector Machines (SVMs) is a popular supervised machine learning algorithm used for binary classification problems. SVM works by finding the maximum margin hyperplane $w * x - b = 0$ in a high-dimensional space that separates the data into classes. This hyperplane is called the decision boundary, and the points closest to the decision boundary are called support vectors.

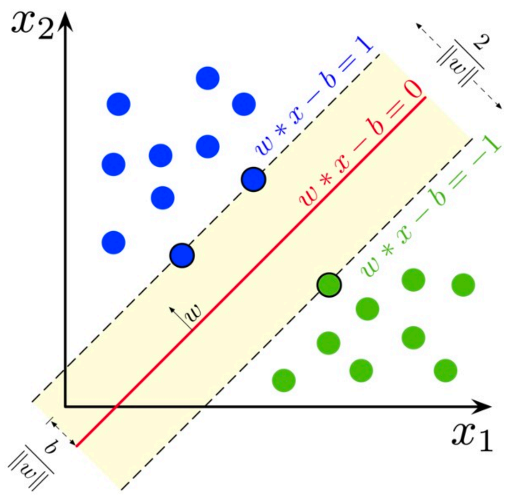


Figure 4

2.5.2 Kernel SVM

One of the key features of SVM is its ability to handle non-linearly separable data through the use of different kernel functions. The kernel functions transform the input data into a higher dimensional space, where it becomes possible to separate the classes linearly. (It means the classes can be separated by a hyperplane.) Even though it is not linearly separable in the lower dimension, as what we observed in the 3D scatter plot above.

For multi-class classification, the SVM is utilized after breaking down the multi-classification problem into multiple binary classification problems. Normally, the One-to-Rest approach is used. In the One-to-Rest approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points.

2.6 Instance-based algorithm:K-nearest neighbors (KNN)

The K-nearest neighbor (KNN) model is a straightforward algorithm that is utilized for solving classification problems. The basic principle behind this model is to classify a new observation based on its proximity to other observations within the training data.

The proximity used is usually the Euclidean distance, which calculates the straight-line distance between two points in a multidimensional space. However, other proximity measures such as Manhattan distance, Minkowski distance, and cosine similarity can also be utilized.

The value of K in the KNN model determines the size of the neighborhood of observations that will be used to make a prediction. For instance, if K=3, then the three nearest neighbors to a new observation will be used to make a prediction. The prediction is the most frequent class among the neighbors.

The KNN model is a lazy learning algorithm, which means that it does not build a model during the training phase, but rather stores the training data. During the prediction phase, the KNN model calculates the distance between the new observation and all the training data, and selects the K nearest neighbors to make a prediction. Therefore, the KNN model can be computationally expensive, especially for large datasets.

3 Relationships

3.1 Decision trees and Random forest

Obviously, the decision trees and random forest are related in that random forest is an ensemble of decision trees.

3.2 Naive Bayes and Logistic regression

1. Suppose that $y_i \in \{-1, +1\}$ and features are multinomial We can show that

$$h(\mathbf{x}) = \underset{y}{\operatorname{argmax}} P(y) \prod_{\alpha-1}^{d} P\left(x_\alpha \mid y\right) = \operatorname{sign}\left(\mathbf{w}^\top \mathbf{x} + b\right)$$

That is,

$$\mathbf{w}^\top \mathbf{x} + b > 0 \Longleftrightarrow h(\mathbf{x}) = +1.$$

As before, we define $P\left(x_\alpha \mid y = +1\right) \propto \theta_{\alpha+}^{x_\alpha}$ and $P(Y = +1) = \pi_+$:

$$\begin{aligned}
[\mathbf{w}]_\alpha &= \log\left(\theta_{\alpha+}\right) - \log\left(\theta_{\alpha-}\right) \\
b &= \log\left(\pi_+\right) - \log\left(\pi_-\right)
\end{aligned}$$

If we use the above to do classification, we can compute for $\mathbf{w}^\top \cdot \mathbf{x} + b$ Simplifying this further leads to

$$
\begin{aligned}
l\mathbf{w}^\top \mathbf{x} + b > 0 &\Longleftrightarrow \sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha \overbrace{\left(\log\left(\theta_{\alpha+}\right) - \log\left(\theta_{\alpha-}\right)\right)}^{[\mathbf{w}]_\alpha} + \overbrace{\log\left(\pi_+\right) - \log\left(\pi_-\right)}^{b} > 0 \\
&\Longleftrightarrow \exp\left(\sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha \left(\log\left(\theta_{\alpha+}\right) - \log\left(\theta_{\alpha-}\right)\right) + \log\left(\pi_+\right) - \log\left(\pi_-\right)\right) > 1 \\
&\Longleftrightarrow \prod_{\alpha=1}^{d} \frac{\exp\left(\log \theta_{\alpha+}^{[\mathbf{x}]_\alpha} + \log\left(\pi_+\right)\right)}{\exp\left(\log \theta_{\alpha-}^{[\mathbf{x}]_\alpha} + \log\left(\pi_-\right)\right)} > 1 \\
&\Longleftrightarrow \prod_{\alpha=1}^{d} \frac{\theta_{\alpha+}^{[\mathbf{x}]_\alpha} \pi_+}{\theta_{\alpha-}^{[\mathbf{x}]_\alpha} \pi_-} > 1 \\
&\Longleftrightarrow \frac{\prod_{\alpha=1}^{d} P\left([\mathbf{x}]_\alpha \mid Y=+1\right) \pi_+}{\prod_{\alpha=1}^{d} P\left([\mathbf{x}]_\alpha \mid Y=-1\right) \pi_-} > 1 \\
&\Longleftrightarrow \frac{P(\mathbf{x} \mid Y=+1)\pi_+}{P(\mathbf{x} \mid Y=-1)\pi_-} > 1 \\
&\Longleftrightarrow \frac{P(Y=+1 \mid \mathbf{x})}{P(Y=-1 \mid \mathbf{x})} > 1 \\
&\Longleftrightarrow P(Y=+1 \mid \mathbf{x}) > P(Y=-1 \mid \mathbf{x}) \\
&\Longleftrightarrow \underset{y}{\operatorname{argmax}} P(Y=y \mid \mathbf{x}) = +1
\end{aligned}
$$

Therefore, the point x lies on the positive side of the hyperplane iff Naive Bayes predicts +1. In the case of continuous features (Gaussian Naive Bayes), when the variance is independent of the class ($\sigma_{\alpha c}$. is identical for all c), we can show that

$$
P(y \mid \mathbf{x}) = \frac{1}{1 + e^{-y(\mathbf{w}^\top \mathbf{x} + b)}}
$$

This model is also known as logistic regression. NB and LR produce asymptotically the same model if the Naive Bayes assumption holds. Visit Naive Bayes and Logistic regression to learn more.

3.3 Logistic regression and Single layered perceptrons

It is perfect to say a logistic regression is a single layered perceptron. Logistic regression would be a specific case of a perceptron. In fact in Andrew Ng's deep learning specialization, an intuition for perceptron is given by beginning with logistic regression.

Actually, the multi-class logistic regression (softmax regression) is a single layered perceptron with multiple outputs.

Neural network is a multi-level classification model, logistic regression and softmax regression can be seen as the simplest neural network. The neural network for binary classification has one output node, but for K

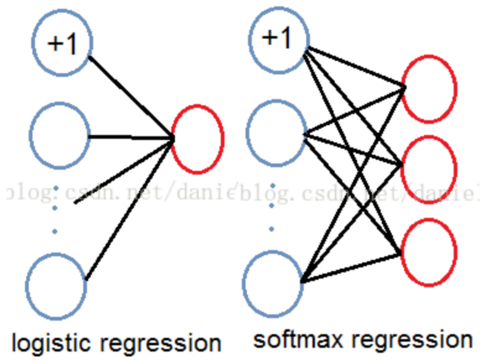


Figure 5: Softmax regression and perceptron

(K>2) classification problems, it has K data nodes. This logistic regression is the same as softmax regression. It is worth noting that the last hidden layer and output layer in the neural network can be viewed as a logistic regression or softmax regression model. The previous layers just learn features from the raw input data and then pass the learned features to logistic regression or softmax regression. Therefore, a neural network for classification can be divided into two parts, feature learning and logistic regression or softmax regression.

3.4 Logistic regression and SVM

For the logistic regression, If instead of using the probability as the threshold, we use the likelihood ratio as the threshold:

$$\frac{P(y=1 \mid x)}{P(y=0 \mid x)} \geq c,$$

for some arbitrary constant $c > 1$. Taking the log of both sides:

$$\log(P(y=1 \mid x)) - \log(P(y=0 \mid x)) \geq \log(c)$$

and plugging in the definition of P:

$$w^T x + b - 0 \geq \log(c) \Longrightarrow w^T x + b \geq \log(c)$$

c is arbitrary, so we pick it to satisfy $\log(c) = 1, w^T x + b \geq 1$.

This gives a feasibility problem (specifically the perceptron problem) which may not have a unique solution. Instead, put a quadratic penalty on the weights to make the solution unique:

$$\begin{aligned}&\min \frac{1}{2}\|w\|^2\\ &\text{s.t. } \ (2y_n - 1)\left(w^T x_n + b\right) \geq 1, \forall n = 1 \ldots N\end{aligned}$$

This gives us an SVM! We derived an SVM by asking logistic regression to use the likelihood ratio as the threshold.

4 Discussion

In the following, we compare each technique. Part of the results of the discussion are summarized in tabular form We hope that the following comments will help you not to choose a completely inappropriate algorithm for their problem.

	Decision Tree	Random Forest	Naive Bayes	Logistic Regression	Neural Networks	SVM	KNN
Accuracy in general	**	****	*	***	***	****	**
Speed of learning withr espect to number of attributes and the number of instances	***	**	****	***	*	*	****
Speed of classification	****	***	****	****	****	****	*
Tolerance to missing values	***	***	****	***	*	**	*
Dealing with discrete/binary/continuous attributes	****	****	***(not continuous)	****	***(not discrete)	**(not discrete)	***(not discrete)
Tolerance to noise	**	***	***	***	**	**	*
Dealing with overfitting	**	**	***	***	*	**	***
Interpretability	****	**	****	****	*	*	**
Transparency	****	***	****	***	*	*	****
Ease of use (the number of parameters to be tuned)	***	**	****	***	*	*	****
Deal with multiple classification problems	****	****	****	**(binary)	****	**(binary)	****

Table 1: Comparing learning algorithms (**** stars represent the best and * star the worst performance)

1. **Accuracy in general:** The accuracy of random forests without pruning can be very high. kernel SVM can also achieve high accuracy on the training set. However, Naive Bayes tends to underfitting, although it usually has better prediction accuracy for new data.

2. **Speed of learning:** KNN, as the lazy learning algorithm, require zero training time because the training instance is simply stored. Naive Bayes methods also train very quickly since they require only a single pass on the data either to count frequencies (for discrete variables) or to compute the probability density function (for continuous variables). Univariate decision trees are also reputed to be quite fast at any rate, several orders of magnitude faster than neural networks and SVMs.

3. **Speed of classification:** KNN classification is very slow because it needs to calculate the distance between the new observation and all points on the training set. This is also a characteristic of the lazy algorithm. The other algorithms are relatively fast, with the exception of random forests, which require a large number of decision trees to make predictions.

4. **Missing values:** Naive Bayes is naturally robust to missing values since these are simply ignored in computing probabilities and hence have no impact on the final decision. On the contrary, KNN and neural networks require complete records to do their work.

5. **Dealing with discrete/binary/continuous attributes:** SVMs and perceptron tend to perform much better when dealing with multi-dimensions and continuous features. On the other hand, decision tree and random forest tend to perform better when dealing with categorical features. The naive Bayes assumes that the input attributes are categorical, meaning that they have a discrete set of possible values.

6. **Tolerance to noise:** KNN is considered intolerant of noise; its similarity measures can be distorted by errors in attribute values, thus leading it to misclassification. Contrary to KNN, decision trees are considered resistant to noise because their pruning strategies avoid overfitting the data in general and noisy data in particular.

7. **Dealing with overfitting:** Naive Bayes is known to be hard to overfit, while Neural Networks are known to be easy to overfit. Decision Tree and Random Forest are also prone to overfitting, but can be solved by pruning. Logistic Regression can also be mitigated by means of variable selection. The overfitting problem of KNN depends on the choice of k value.

8. **Interpretability *(whether the prediction procedure of the method is easily understood)*:** Naive Bayes and logistic regression are all considered very easy to interpret, whereas neural networks and SVMs have notoriously poor interpretability. KNN is also considered to have very poor interpretability because an unstructured collection of training instances is far from readable, especially if there are many of them.

9. **Transparency *(whether the principle of the method is easily understood)*:** KNN is quite transparent (while the resulting classifier is not quite interpretable) because it appeals to the intuition of human users, who spontaneously reason in a similar manner. Similarly, Naive Bayes and decision tree are very transparent, as it is easily grasped by users like physicians who find that these algorithms replicate their way of diagnosing. However, the neural network and kernel SVMs are hard to be comprehended.

10. **Ease of use (the number of parameters to be tuned):** Neural networks and SVMs have more parameters than the other techniques. The basic kNN has usually only a single parameter k which is relatively easy to tune.

11. **Deal with multiple classification problems:** SVMs and logistic regression are binary algorithms by default. To apply them to multiple classification problems, additional operations are required. But other models are directly applicable to multiple classification problems.

12. **High-variance and high-bias algorithm:** A high-variance algorithm has the ability to generate complex models that fit the training data very closely but prone to overfitting, meaning they may not generalize well to new data. Examples of high-variance algorithms include decision trees, random forest, neural networks, and SVMs.

A high-bias algorithm, tends to generate simple models with low capacity to learn complex patterns. Such models are less likely to overfit the training data and may generalize well to new data but may result in lower accuracy on the training data. Examples of high-bias algorithms include Naive Bayes and logistic regression.

13. **Decision boundary:** The decision tree, logistic regression and naive Bayes have linear decision boundary. The neural networks, KNN and kernel SVMs have non-linear decision boundary.

14. **Storage space:** The basic KNN algorithm uses a great deal of storage space for the training phase, and its execution space is at least as big as its training space. But the naive Bayes requires little storage space during both the training and classification stages: memory needed to store the prior and conditional probabilities. On the contrary, for all eager learners, execution space is usually much smaller than training space, since the resulting classifier is usually a highly condensed summary of the data.

In conclusion, no single classification algorithm consistently outperforms other algorithms across all data sets. When faced with the question "Which algorithm should we use to model our classification problem?". One should carefully do exploratory data analysis (as we did multiple times in our tutorials), and then make decisions based on the characteristics of the data set and in combination with the above discussion.