Methods in Open Science

Machine Learning

Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

WV VIEW

AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

Now that we have discussed linear regression and associated generalized linear models, we will move on to discuss other methods for predicting a nominal or numeric variable that fall under the general umbrella of machine learning.

Machine Learning

- Nonparametric
- Learn from data
- Perform classification and regression
- Deal with complex feature space, relationships, patterns, and noisy data
- A wide variety of methods available

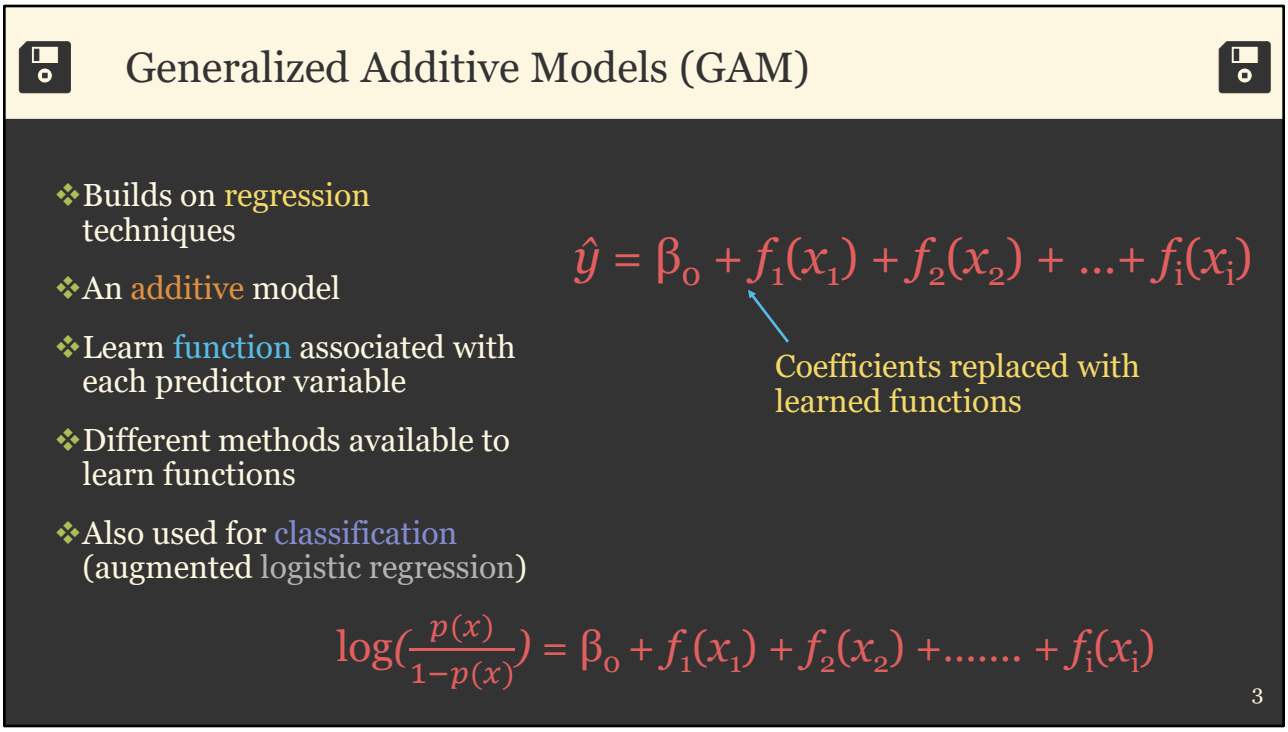
We will explore:

1. Generalized Additive Models (GAM)
2. *k*-Nearest Neighbor (*k*NN)
3. Decision Trees (DT)
4. Random Forest (RF)
5. Boosted Decision Trees (Boosted DT)
6. Explainable Boosting Machines (EBM)
7. Support Vector Machines (SVM)
8. Artificial Neural Networks (ANN)

2

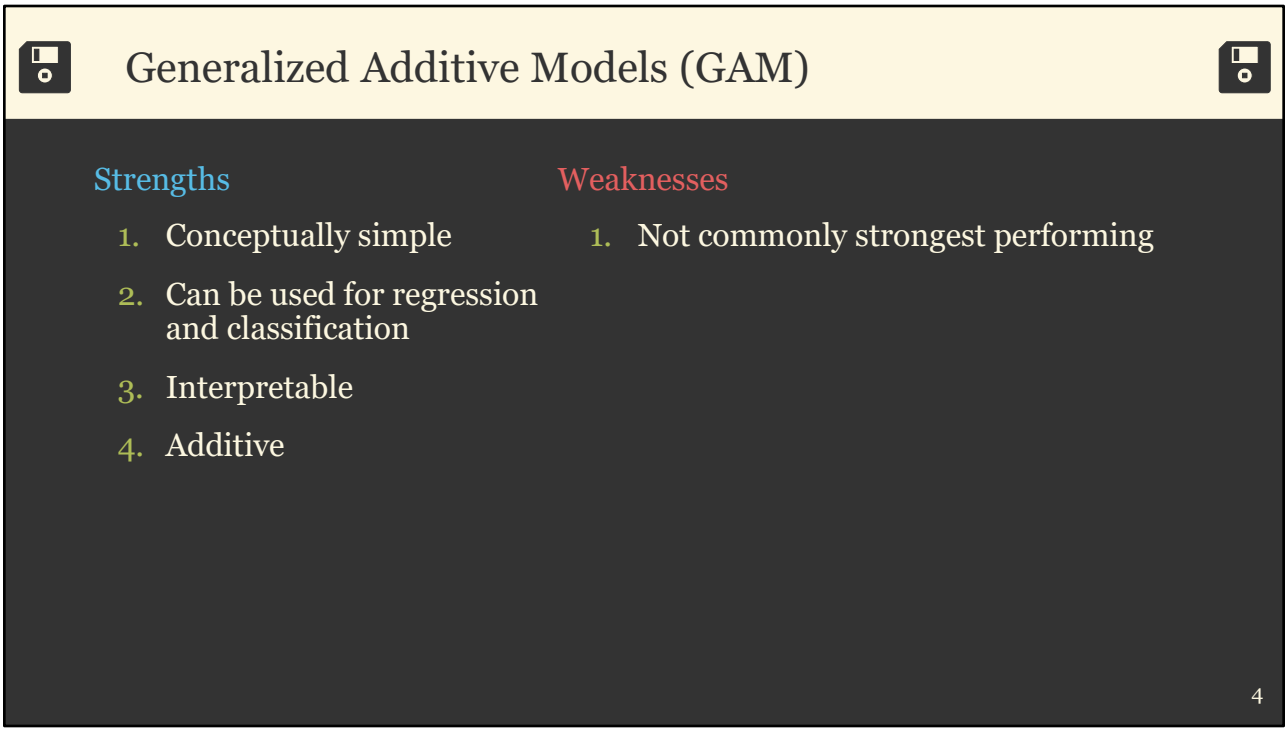
This slide highlights the key characteristics of machine learning methods. These methods are nonparametric, or do not have distribution assumptions. They are often used in a supervised manor, so they require training data. The models tend to be more robust to complex patterns and relationships than parametric methods. It is also possible to incorporate a variety of predictor variables to potentially improve the prediction.

In the following slides, we will discuss the methods listed on this slide. Note that these are not the only options and that there are a lot of algorithms available with new methods being developed as the field progresses. I chose these methods as they are some of the most commonly used.



Generalized additive models, or GAMs, offer an extension of linear regression methods. In contrast to linear and multiple linear regression, GAMs do not assume a linear relationship between predictor variables and the response variable. Instead, relationships are modeled using smoothing, spline, or other methods. A response variable is predicted by learning a y-intercept along with functions that describe the relationship between the response and each predictor variable. Essentially, the coefficients in a multiple linear regression model are replaced with learned functions that are not confined to a linear relationship. The model is additive because separate functions are learned for each predictor variable independently, which allows for an examination of the effect of each predictor variable separately.

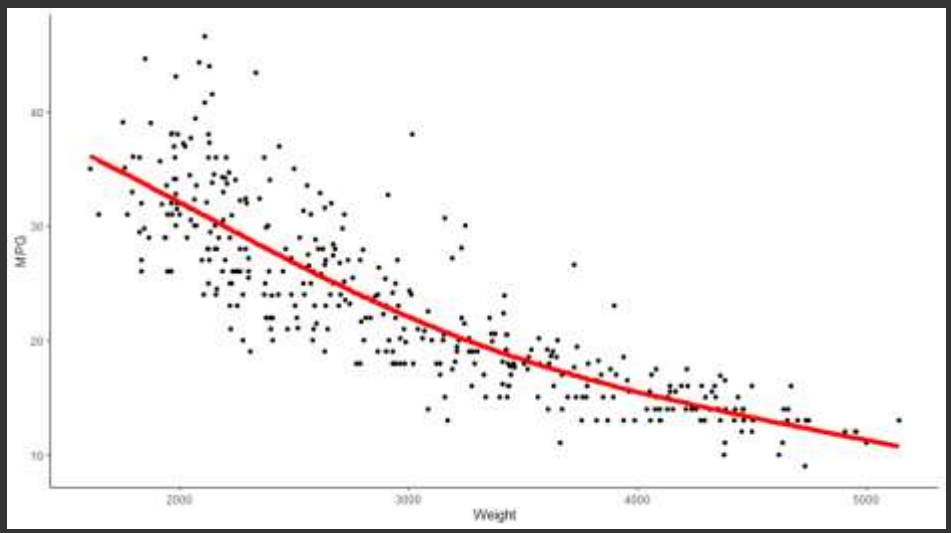
In order to apply GAMs to binary classification problems, class logits are predicted as opposed to a continuous variable, essentially generalizing logistic regression. On the slide, the top equation represents a regression problem, and the bottom equation represents a binary classification problem. Again, there are multiple methods that are available to generate the functions associated with each predictor variable.



GAMs are conceptually simple, similar to linear regression. They can be applied to both regression and binary classification problems, are interpretable, as the user just need to visualize the learned functions associated with each predictor and are additive. The additive nature makes it easier to understand the relationship between each predictor variable and the dependent variable.

I have generally found that GAMs are not the strongest performers in regards to the accuracy obtained. However, this is not true in all cases, and some decrease in accuracy may be tolerated to obtain increased interpretability. Later in the module, we will explore a recent adaptation of GAMs, known as Explainable Boosting Machines, that has been documented to improve the accuracy of more traditional GAMs.

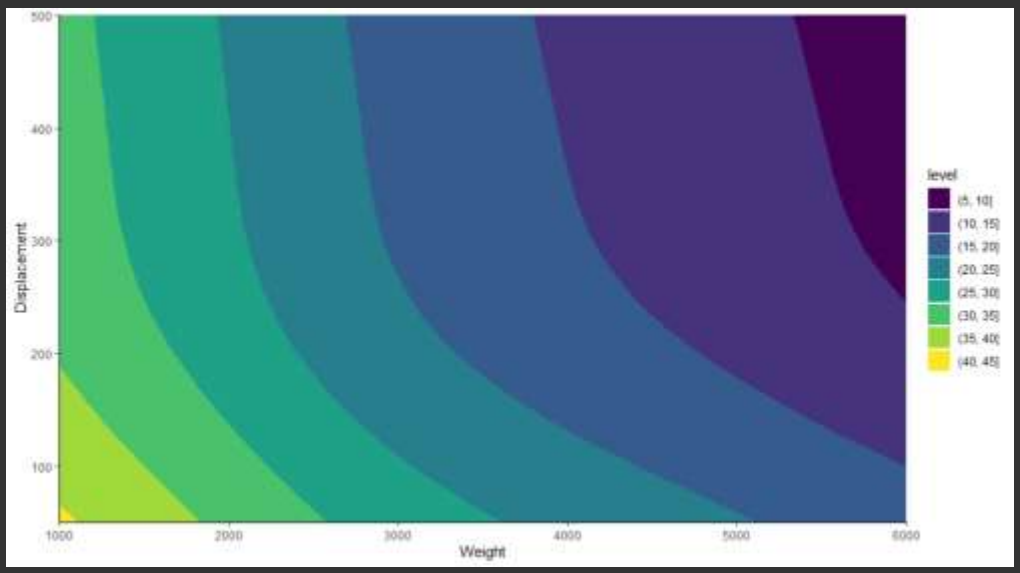
Example Data: Auto MPG



5

This result shows a model, the red line, generated using GAMs that estimates fuel efficiency for the Auto MPG data using vehicle weight. The modeled relationship is no longer linear and seems to better capture the relationship between the variables.

Example Data: Auto MPG

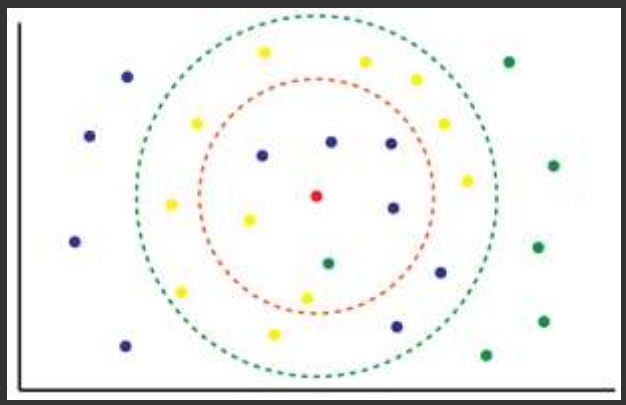


6

This image represents the GAM when using engine displacement and vehicle weight to predict fuel efficiency. The color ramp indicates the predicted fuel efficiency. Again, the model is no longer linear since coefficients have been replaced with learned functions.

k-Nearest Neighbors (*k*NN)

- Compare new sample to nearest training sample(s) in the feature space
- *k* = number of neighbors compared to
- Assign new sample to majority of the neighbors (classification)
- Use average or distance weighted average of neighbors (regression)
- Proportion of neighbors by class (probability)



7

k-NN is conceptually simple. Let's begin by discussing this method in the context of a classification problem.

The basic idea here is that a new observation is assigned to the class of the nearest observations in the feature space. In the provided image, the feature space is two dimensional (e.g., two predictor variables), but distance measures can be extended into multiple dimensions mathematically. The number of neighbors to consider is based on the *k* parameter.

In the example, the red dot represents a new sample for which an inference need to be made. The red ring represents 7 nearest neighbors. Of the 7, 4 are blue, 2 are yellow, and 1 is green. So, the new sample would get assigned to the blue class.

The green ring represent 18 nearest neighbors. From these neighbors, the yellow class is most abundant, so the new sample would get mapped to that class.

k-NN can also be used to predict continuous variables (regression) based on the average or inverse distance weighted average of the nearest neighbors. Inverse distance weighted averaging places more weight on features closer to the new sample, or the weight of each neighbor in the calculation is weighted

by the inverse of its distance form the data point being predicted.

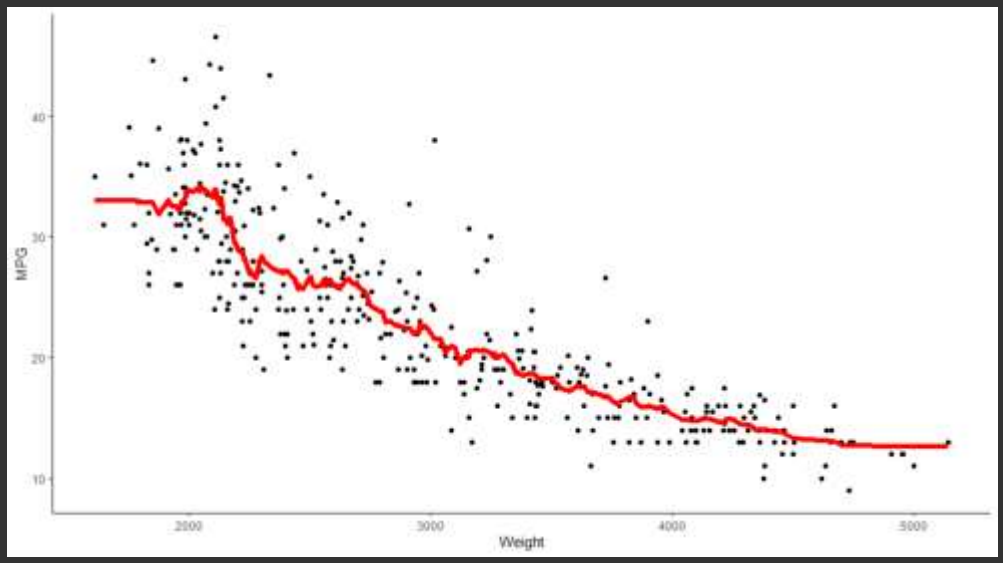
For class probabilities, the proportion of each class in the set of neighbors is used.

The results of the model will vary based on the value used for *k*. This is an example of a hyperparameter that is not learned in the modeling process but must be set by the user. This parameter is often optimized. We will discuss optimization methods in a later module. Generally, lower values of *k* may result in overfitting to the training data while larger values can result in overgeneralization. As discussed above, optimizing algorithms is often a balance between overfitting and overgeneralization.

Another important consideration for *k*-NN is that all predictor variables must be rescaled since the prediction is based on calculations of distance in the multidimensional feature space. It is possible to incorporate categorical predictor variables; however, different distance metrics are generally used. There are a variety of different distance metrics available. The simplest is Euclidean, or straight-line distance, in the feature space. In a two- or three-dimensional space we can visualize this distance as a line. Although we cannot easily visualize this Euclidean distance in a larger feature space, it can be mathematically described. A full discussion of measures of distance that can be used within *k*-NN models is outside the scope of this course.

Another unique aspect of *k*-NN is that the model is not trained in the traditional since. Instead, in order to perform inference, each new observation is compared to the training samples to which it is closest to in the multidimensional feature space.

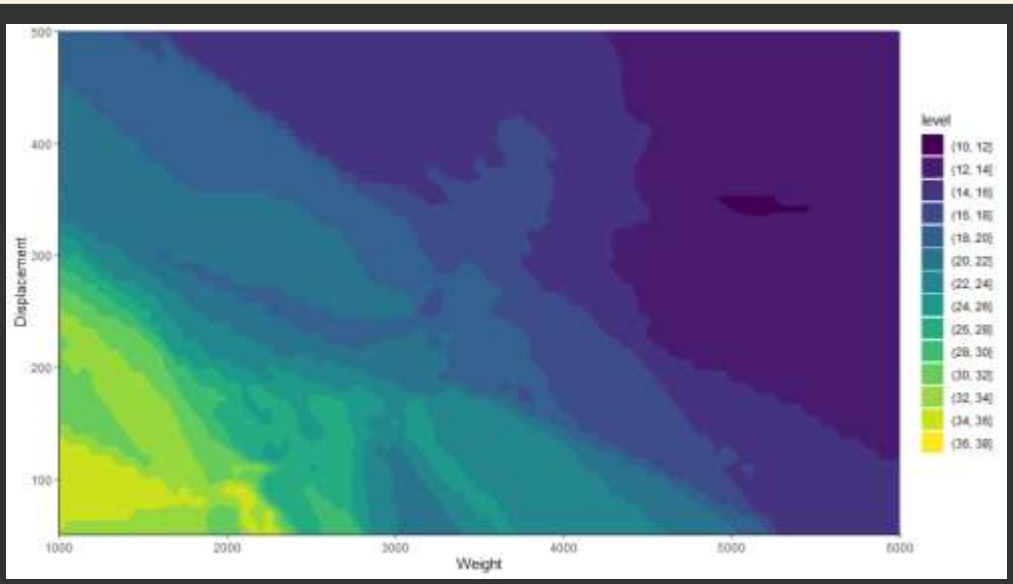
Example Data: Auto MPG



8

This is an example of a kNN model for predicting fuel efficiency using vehicle weight.

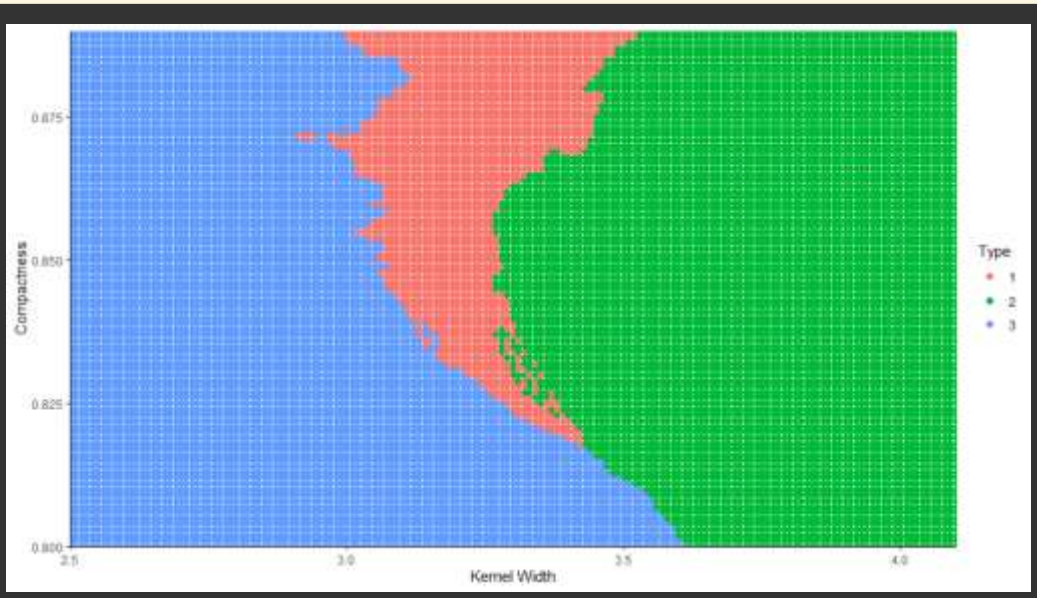
Example Data: Auto MPG



9

This image visualizes the fuel efficiency model using the vehicle weight and engine displacement variables.

Example Data: Seeds



10

This represents the results for the Seeds data in which the three varieties are differentiated using kernel width and kernel compactness. 1 = Kama, 2 = Rosa, and 3 = Canadian.

k-Nearest Neighbors (*k*-NN)

Strengths

1. Nonparametric
2. Conceptually simple
3. Consistent
4. Can be used for regression and classification

Weaknesses

1. Can be slow
2. Computationally intensive for larger training sets
3. Negatively impacted by large feature space
4. Negatively impacted by noisy or irrelevant features
5. Need consistent scale across variables
6. Generally, I have found this algorithm to rarely be the optimal choice

11

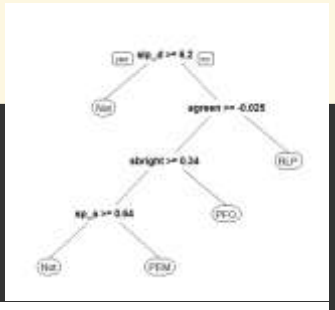
k-NNs' strengths include that it is nonparametric, conceptually simple, and consistent (i.e., not stochastic) when given the same set of training samples. It can be used for regression, binary classification, multiclass classification, and probabilistic prediction.

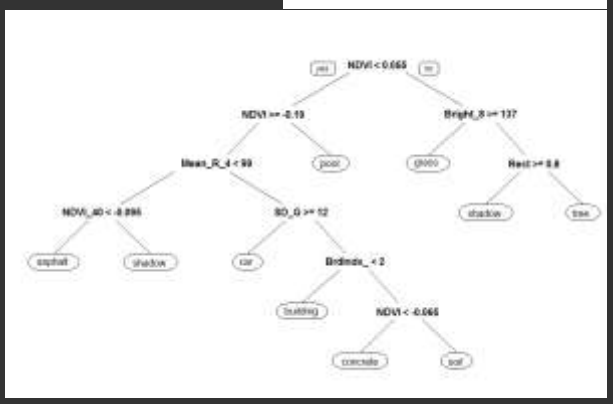
Some weaknesses are that it can be slow since each new sample must be compared to the nearest training samples in the feature space and that it is computationally intense for larger training sets. It can be negatively impacted by large feature spaces (i.e., the Hughes Phenomenon) due to the added complexity of the feature space. It also tends to be negatively impacted by the inclusion of noisy or irrelevant features; as a result, feature selection is often implemented before using *k*-NN. We will discuss feature selection in a later module. Lastly, it is sensitive to the scale of the predictor variables, so it is necessary to rescale all predictor variables to a common scale, such as from 0 to 1. This is because the decisions are based on distances in the multidimensional feature space.

Generally, I find that this algorithm is often not the optimal choice in terms of accuracy.

Decision Trees (DT)

- Recursively partition the data into more homogeneous subsets
- Binary recursive partitioning
- Based on measure of refinement
- Final nodes = leaf nodes
- Overfit to the training data
- Needs pruning





12

Decision trees are amongst the most intuitively simple machine learning methods. The goal here is to split the data into more homogenous subsets to create a series of decision rules that can then be used to obtain a prediction. I often think of decision trees as a dichotomous key, such as those used to identify the species of a tree. A series of yes-or-no questions are posed that eventually lead to a final prediction. The final predictions in the tree structure are called leaf nodes. Note also that each split is commonly based on a single variable and is binary: yes-or-no. Sequences of rules are combined to perform recursive partitioning of the data.

One issue with a fully grown decision tree is that it tends to overfit to the training data. In order to combat overfitting, trees are often pruned, which consists of removing some of the later splits. Thus, pruning is the most commonly used method to combat overfitting and improve model generalization for decision trees.

Why Prune?

- Overfit to the training data
- Need pruning

Evaluation Data Set	Before pruning		After pruning	
	Tree size	Errors (%)	Tree size	Errors (%)
Training data	713	1.6	231	8.6
Test data	713	17.6	231	15.7

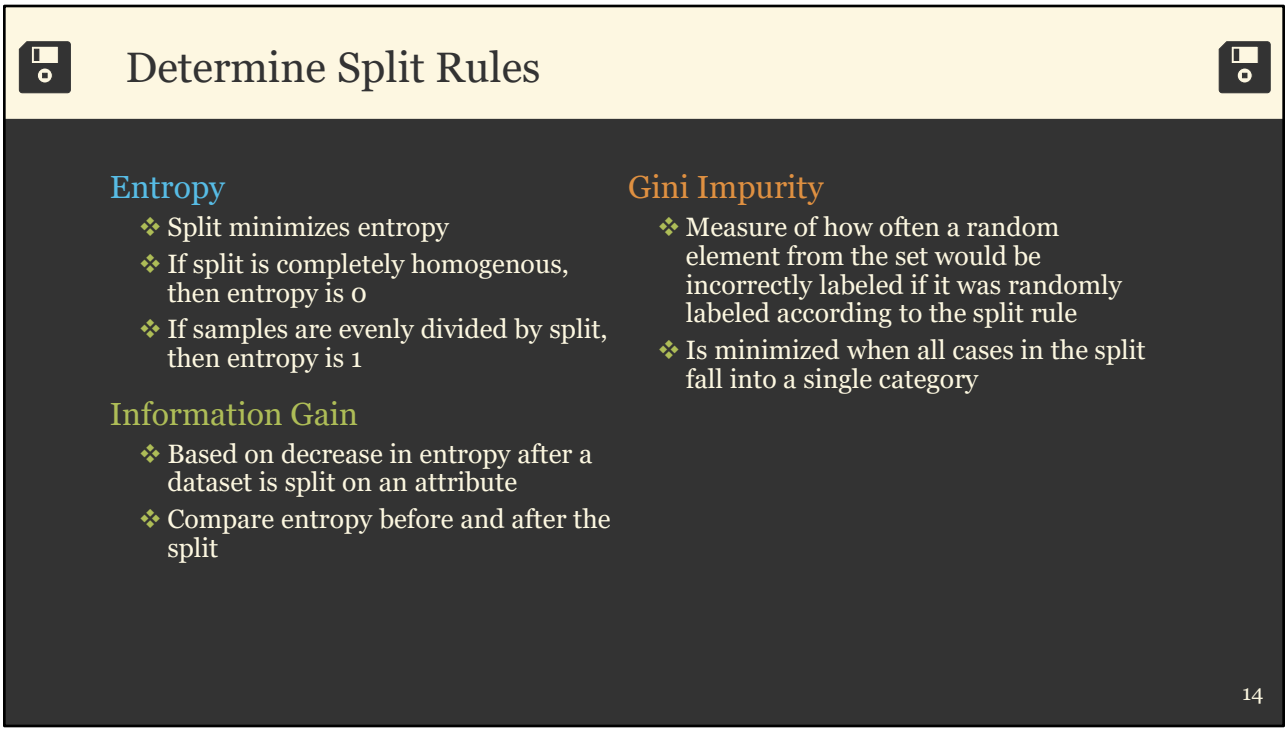
Example provided by Dr. Tim Warner

13

The example on this slide, provided by Dr. Tim Warner, highlights the use and purpose of pruning. The before pruning results represent the error rates for the training and testing data when the tree is grown to its full depth and not pruned. The error rate for the training data is very low (1.6%) while the error rate for the test data, or new samples, is much higher (17.6%). The tree consists of a total of 713 split rules.

Once the tree is pruned to include fewer split rules, 231, the error rate for the training data increases to 8.6% while the error rate for the test data improved to 15.7%. This suggests that the model is now less overfit to the training data and generalizes better to the new samples.

Similar to the *k* parameter for *k*-NN, the pruning parameter, often termed the complexity parameter, is not learned during the training process. It is a hyperparameter that can be tuned to potentially improve performance.

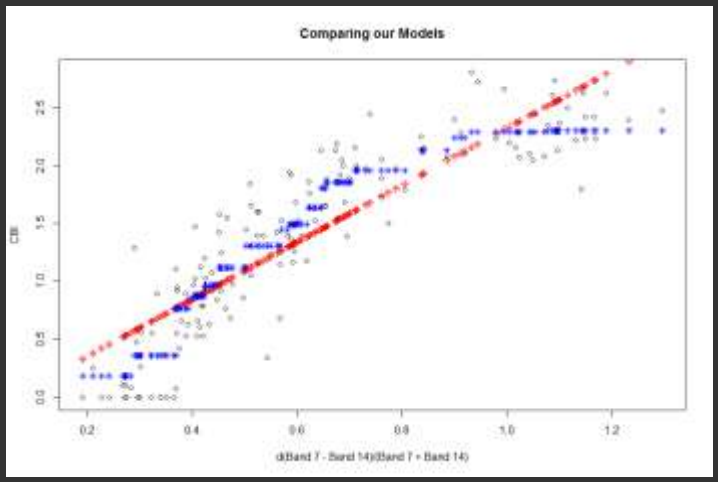


How is the optimal split at a certain node determined? This is generally based on some metric, and several options are available. Entropy relates to homogeneity within partitions. If a split is completely homogenous in that all samples from the same class fall on one side of the decision boundary, the entropy relative to that class will be zero. This indicates a good splitting rule for that class. In contrast, if the samples from one class are evenly partitioned by the decision rule, the entropy would be 1 relative to that class, suggesting a poor splitting rule. Information gain expands upon entropy by assessing the entropy before and after a split is made.

Another option is the Gini Impurity metric, which is a measure of how often a random element from the set would be incorrectly labeled if it were labeled according to the split rule. This measure is minimized when all cases from a class fall on one side of the partition.

DT for Regression

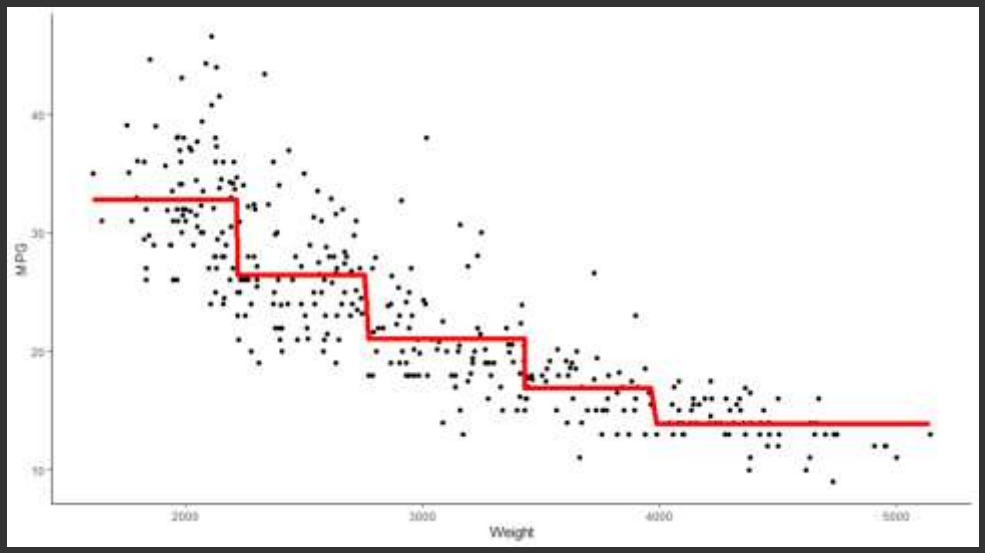
- DT can be used for regression
- However, each leaf node will not be a single value but a range of values



15

Decision trees can also be used for regression problems. Here each leaf node will predict a range of values as opposed to a class. When using a single variable, this results in a stair-step pattern where each step is associated with a leaf node. In the example graph, the red symbols represent a linear model while the blue symbols represent the decision tree model.

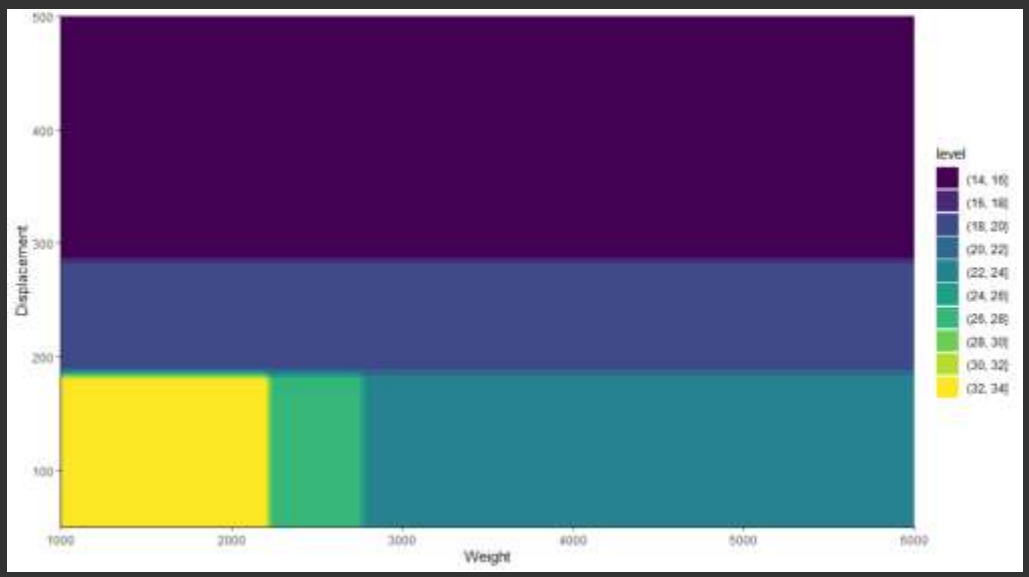
Example Data: Auto MPG



16

This slide provides another example of a regression output for predicting fuel efficiency using vehicle weight. Again, note the stair-step pattern obtained which relates to the different leaf nodes in the tree.

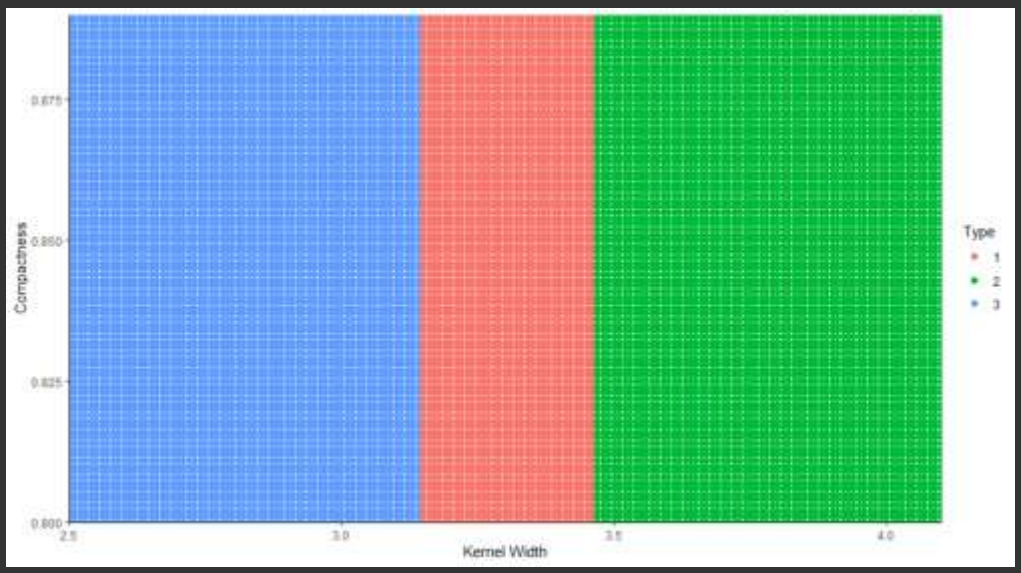
Example Data: Auto MPG



17

This graphic represents a decision tree model in a two-dimensional space. Since only a single variable is considered at each splitting node, the result is breaking the two-dimensional space into rectangular regions that are orthogonal to the axes or predictor variables. Each region would correspond to a certain leaf node.

Example Data: Seeds



18

This is an example for the Seeds classification problem. In this example, only the kernel width was used in the final tree, so the compactness is actually not being considered in the decision. Effectively, the model is just ranges of kernel width associated with each of the three varieties.

Decision Trees

Strengths

1. Nonparametric
2. Easy to interpret (to a point)
3. Few user-defined parameters
4. Can be used for regression and classification
5. Predictor variables of different scales okay
6. Can handle numeric and categorical predictor variables
7. Generally quick
8. Works well with large datasets
9. Some built-in feature selection

Weaknesses

1. High variance
2. Not best accuracies generally

19

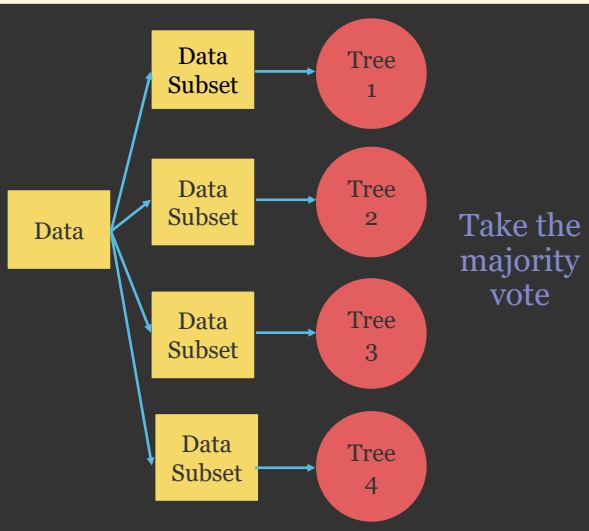
Decision trees have several strengths. They are nonparametric, can be interpretable (depending on the complexity of the tree structure), have few user-defined parameters (i.e., hyperparameters that must be optimized), can be used to perform regression and classification, can accepted a wide range of numeric predictor variables measured on different scales, can incorporate nominal predictor variables without the need to convert them to dummy variables, are generally quick to train and when used for inference, and can handle large training sets. Decision trees also haves some built-in feature selection because weak predictor variables may not be selected to generate splitting rules and will thus not be included in the final model.

Some issues with decision trees include that the resulting tree structure and associated predictions can vary greatly with changes in the training data (i.e., high variance). I have generally found that a single decision tree is not an optimal choice in terms of model accuracy.

Random Forests (RF)

- Uses decision trees
- Uses the Gini Index of Impurity
- Ensemble decision tree method
- Uses random subset of predictor variables for splitting at each node
- Uses random subset of training data in each tree
- Attempts to reduce correlation between trees
- Ensemble of weak classifiers

Breiman, L., 2001. Random forests. *Machine learning*, *45*(1), pp.5-32.



20

One means to potentially expand upon single decision trees is to generate multiple decision trees that act collectively as a single model. Such methods are known as ensemble decision tree methods. These methods have generally been shown to offer strong performance and improvement over single decision trees. There are different methods available to generate an ensemble of decision trees. One option is Random Forest, which is currently very popular.

RF traditionally makes use of the Gini Impurity metric to determine splitting rules; however, augmentations of RF may use different measures. Each tree in the ensemble uses a subset of the training samples, which are selected using bootstrapping, or random sampling with replacement. Also, only a subset of the predictor variables is available for splitting at each decision node. The goal of using a subset of the training data and variables is to reduce the correlation between trees and minimize overfitting, or a set of weak classifiers are collectively strong and generalize well due to reduced overfitting. The method requires the user to determine the number of trees to include in the ensemble. However, the algorithm does not overfit if a large number of trees is used, so it is common to use a large number by default, such as 500 or 1,000. The number of predictor variables available for splitting at each decision node, generally termed *mtry*, is commonly optimized; however, the default settings may provide an adequate result. For classification problems, the final label is determined based on a majority vote or the class that has the largest number of

votes aggregated across all of the trees will be the resulting classification. The distribution or proportion of votes for each class can be used to obtain a probabilistic class prediction. For regression, each node will represent a range of values which can be averaged across trees to obtain the final continuous prediction.

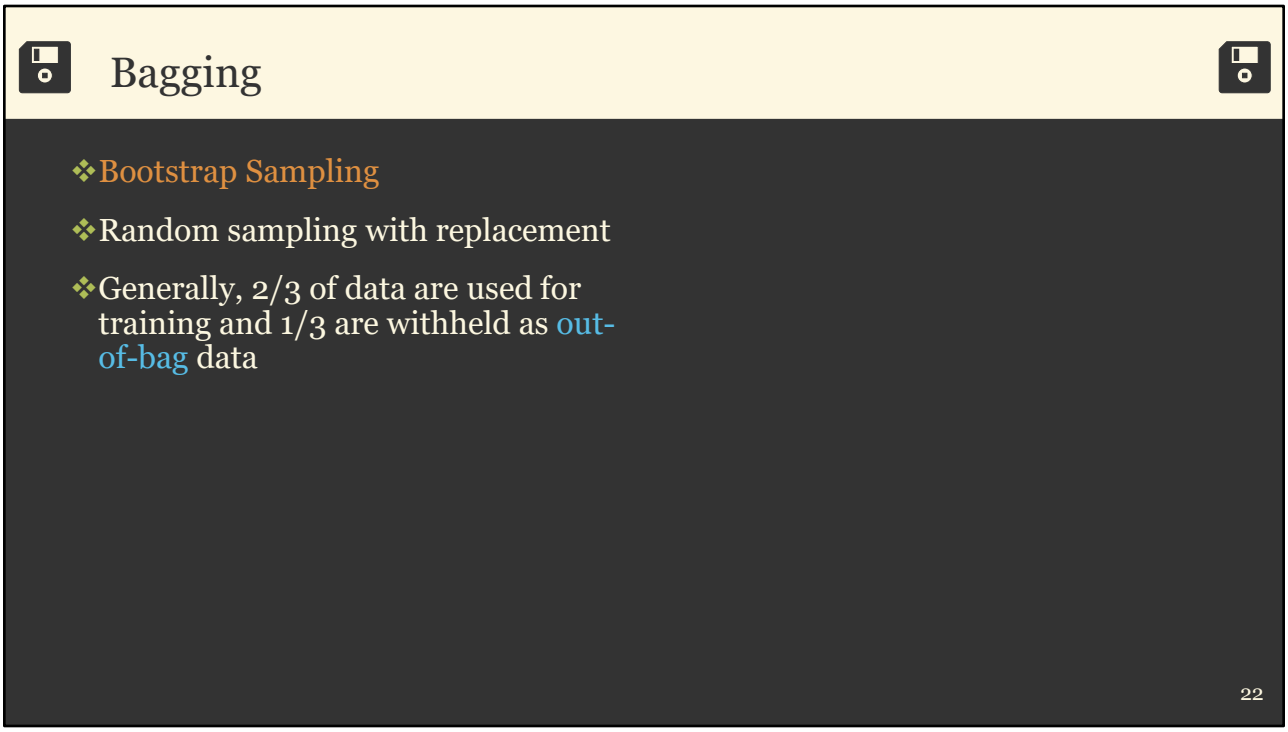
Another interesting feature of random forest is that some internal self assessment is possible since not all training samples are used in each tree. The withheld samples in each tree are known as out-of-bag (OOB) samples. Predicting the OOB samples and aggregating across trees can provide an assessment of model performance.

What does RF use to determine splits?

- Gini Index of Impurity
- Measure of impurity of one class vs. the other classes
- High value means more impure (bad split)
- Low value means more pure (good split)

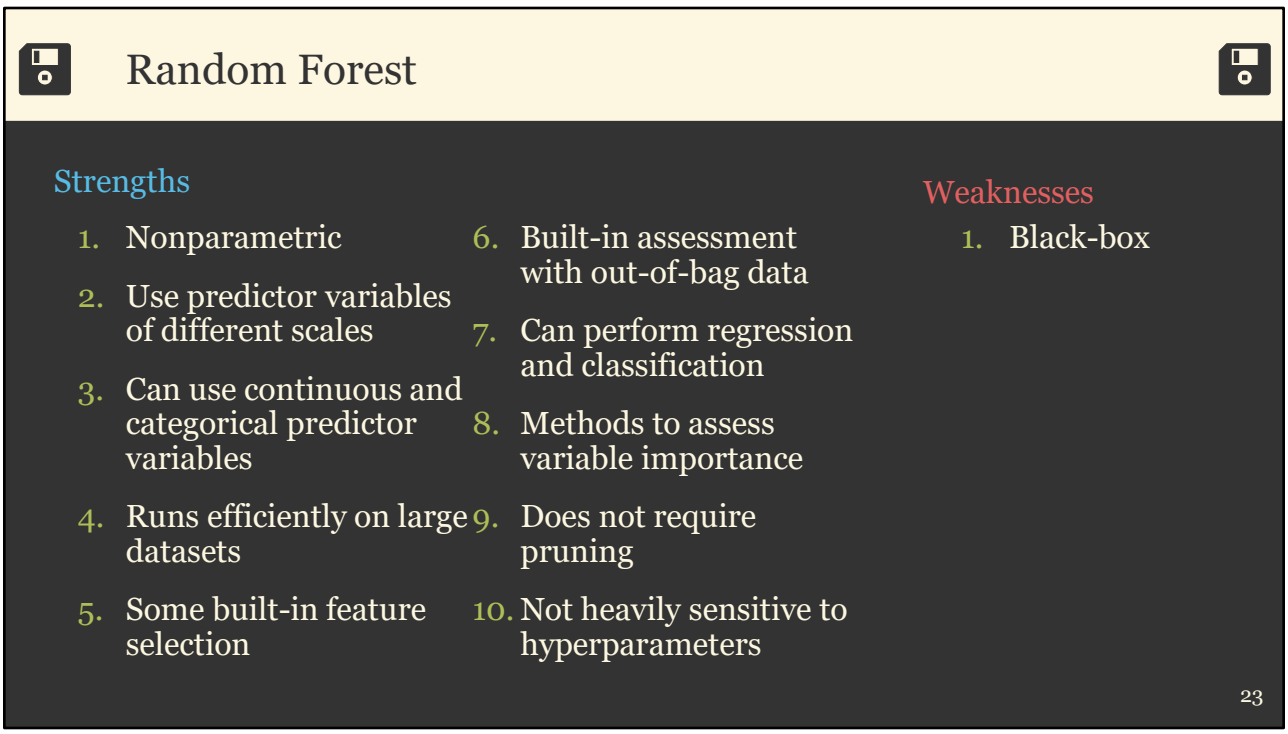
21

As already mentioned, RF traditionally uses the Gini Index of Impurity to determine the split rules at each node. However, some augmentations exist that use other methods.



Bagging, which is implemented by Random Forests, is random sampling with replacement. For example, a random 2/3 of the data could be used to build a tree while the remaining 1/3 are reserved as an OOB sample. Again, the goal of using bagging is to reduce the correlation between trees by allowing each tree to only see a subset of the training samples.

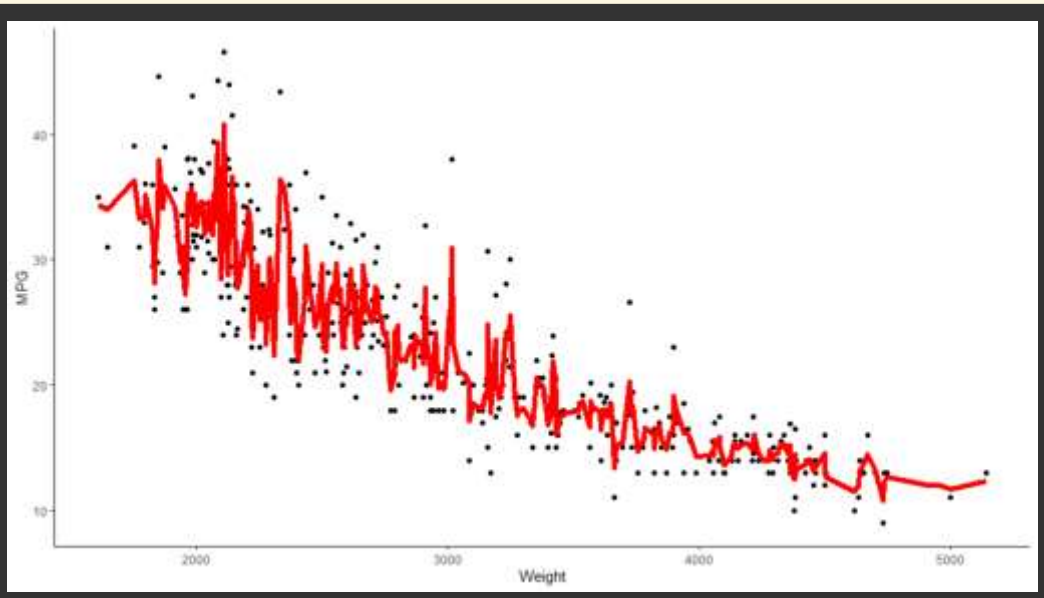
I find it useful to think of bagging as selecting a sample of marbles from a bag of marbles. Before generating a tree, a random subset of marbles is drawn from the bag. The remaining marbles then serve as the OOB sample. Once a tree is produced, all marbles are returned to the bag, it is shook up to encourage randomization, and a new sample is drawn that will likely include a mix of samples used in the prior trees and samples withheld in the prior trees; the makeup of the training samples used in the tree and the withheld OOB samples will be different.



Random Forest has several strengths including being nonparametric and being able to use predictor variables measured on different scales, both numeric and nominal predictor variables, and nominal predictor variables without creating dummy variables. RF runs efficiently on large datasets and has some built-in feature selection since weak predictor variables may not be selected for splitting. It has a built-in assessment method based on predicting the OOB samples not use to generate each tree. It can be used to perform regression, classification, and probabilistic prediction. It is able to provide an assessment of variable importance or contribution within the model. In contrast to single DTs, trees in an RF model do not require pruning as a result of combining multiple trees to generate the final prediction and using a subset of the training samples in each tree and a subset of the predictor variables to select from for splitting at each node. It has been suggested that the method is not highly sensitive to its hyperparameters; however, I generally suggest optimizing the hyperparameters just-in-case.

The major drawback with RF is that it is a black-box. With a large number of trees, it is not possible to conceptualize or visualize the decisions made to generate a prediction.

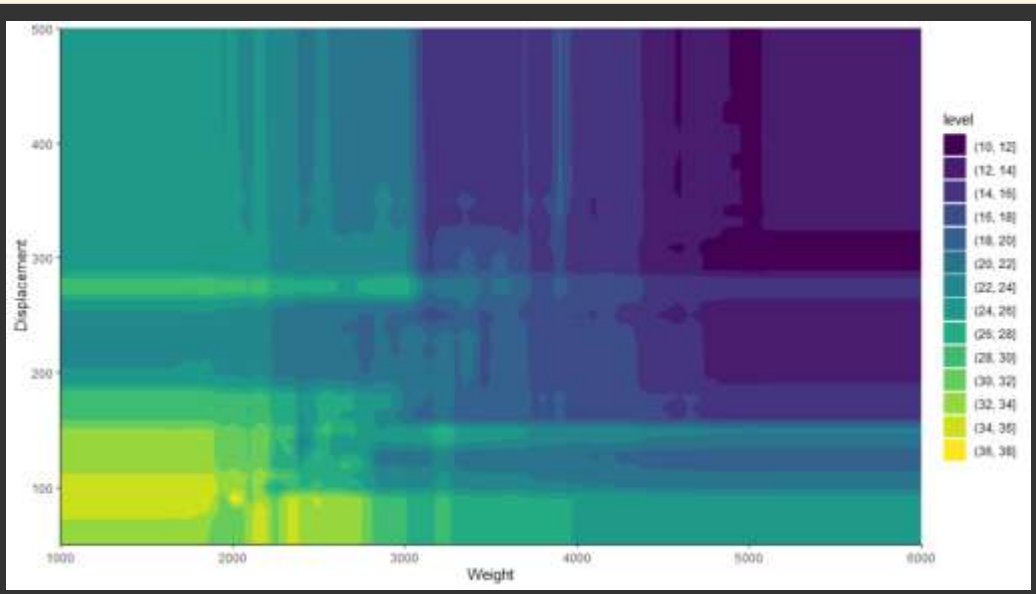
Example Data: Auto MPG



24

The red trend on this graph represents the RF model for predicting fuel efficiency using vehicle weight.

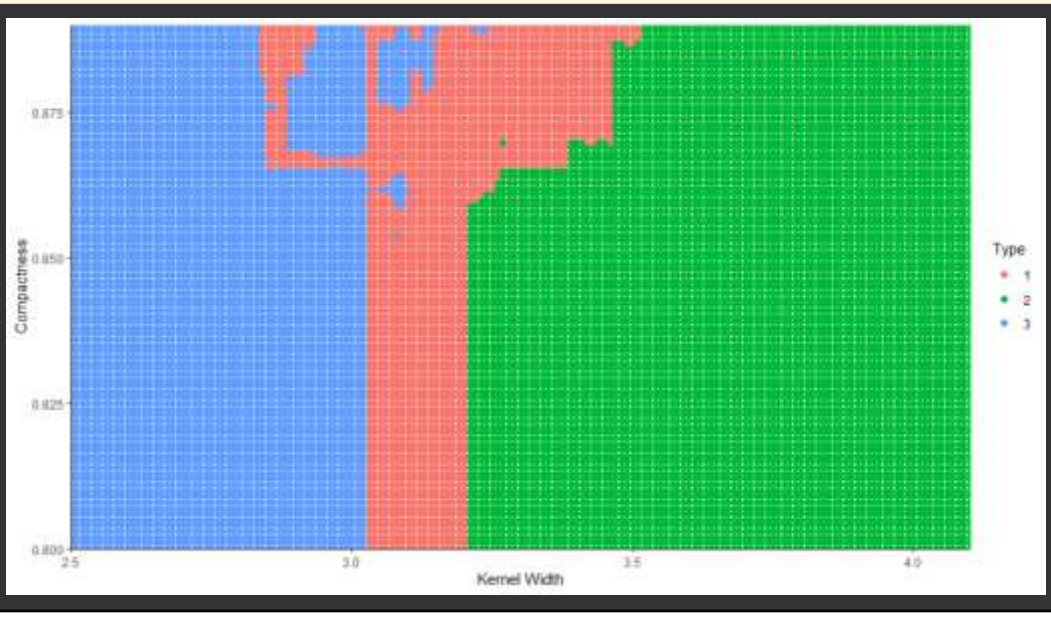
Example Data: Auto MPG



25

This graphic visualizes the result for estimating fuel efficiency using both vehicle weight and engine displacement. Although the model is more complex than that for a single decision tree, similar to a single decision tree each of the decision boundaries are orthogonal to the axes since only one variable is used at each decision node.

Example Data: Seeds



26

This result shows the RF model for differentiating the wheat varieties. Again, note the orthogonal boundaries. Also, in contrast to the DT model shown above, both kernel width and compactness are being used to make the prediction.

Conditional Inference Trees

- Unbiased recursive partitioning
- Uses correlation to make splits (p-values) as opposed to Gini Index
- Permutation-based significance tests
- Not as biased towards nominal variables with large number of levels and correlation or variability in measurement scale for continuous variables.

27

As mentioned above, there are multiple augmentations of the original Random Forest algorithm. One issue that has been noted in regards to the original implementation is that there is a bias in selecting certain variables for splitting. Specifically, nominal variables with a larger number of levels and correlated predictor variables may take on higher weight in the model. There are also some biases associated with difference in measurement scales.

To at least partially alleviate some of these issues, conditional inference trees have been developed. The key difference between traditional RF and conditional inference trees is that variable selection at splitting nodes and splitting rules are based on measures of correlation as opposed to the Gini Index of Impurity.

A full discussion of conditional inference trees is outside the scope of this class. However, I did want to mention this method as an example of an augmentation of the traditional RF algorithm.

Boosted Decision Trees

- Another way to generate an ensemble of trees
- An iterative process
- Samples incorrectly classified in prior trees are given a higher weight in subsequent trees
- Different methods are available to build the trees

28

There are other ways to generate an ensemble of decision trees other than bagging or RF. One option is to implement a boosting method. The key concept behind boosting is that trees are iteratively generated such that subsequent trees focus on fixing misclassifications or errors made in prior trees. This is generally accomplished by weighting the samples that were incorrectly predicted higher in subsequent trees. There are actually a wide variety of methods available to perform this iterative learning process, and a full discussion of these methods is outside the scope of this course.

I have generally found that Boosted Decision Trees and Random Forests yield similar accuracies. However, this may not be true for all problems. Generally, it is best to experiment with multiple algorithms and assess the output using withheld samples in order to determine which model is offering the best performance for a specific problem.

One issue with boosted decision trees is that the process can be slow since it is iterative. This is in contrast to RF, in which each tree is independent of the other trees, and trees can be built in parallel as opposed to sequentially. However, modern computers are generally capable of training boosted decision tree models without issue. You might just have to wait a little longer.

Explainable Boosting Machines (EBM)

- Combination of Generalized Additive Models (GAMs) and Boosted Decision Trees
- An additive method
- Functions associated with each variable are built using boosted decision trees and gradient descent
- Model learns from predictor in a round-robin fashion over many iterations using low learning rate
- Can incorporate interactions
- Goal is to obtain strong predictive performance while also maintaining interpretability

https://interpret.ml/

Lou, Y., Caruana, R., Gehrke, J. and Hooker, G., 2013, August. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 623-631).

29

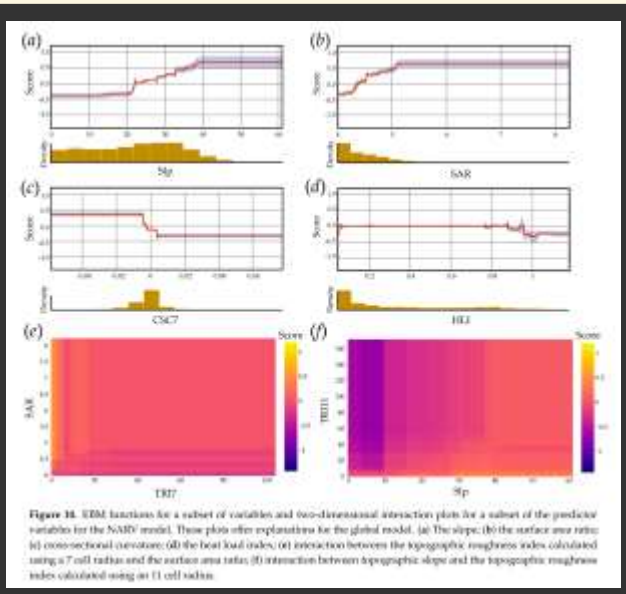
Explainable Boosting Machines (EBMs) are a recently introduced modeling method that combines generalized additive models (GAMs) and Boosted Decision Trees. Remember that GAMs are a generalization of linear models in which the coefficients are replaced with learned functions. In the specific case of EBMs, the functions are learned using Boosted Decision Trees. EBMs have been proposed as a method that offers interpretability and also strong predictive performance in many use cases.

Although GAM equations are more interpretable than models generated using black-box machine learning methods, which cannot be presented as a single equation or set of functions that describe the estimated relationship between the dependent variable and values of each predictor variable, they are often less accurate. EBM expands upon GAMs to maintain interpretability but improve predictive performance. EBM is a fast implementation of the generalized additive models plus interactions (GA^2M) method. Using GA^2M, the function associated with each predictor variable is approximated using many shallow decision trees created with gradient boosting to iteratively improve model performance. More specifically, shallow decision tree generation, learning, and gradient updates are performed using a single predictor variable at a time in a round-robin fashion with a low learning rate. Due to the low learning rate, only small updates to the model are made with the addition of each tree. This requires the model to be built by iterating

through the training data over thousands of boosting iterations in which each tree only use one predictor variable. The algorithm developers argue that the low learning rate reduces the influence of the order in which features are used while iteratively cycling through the predictor variables using a round-robin method minimizes the impact of multicollinearity to maintain interpretability. To take into account interactions between predictor variables, two-dimensional functions can be learned to relate the response variable to pairs of predictor variables. Adding interaction terms requires that the additive nature of GAMs be relaxed. Further, interpreting the influence of a single predictor variable will require investigating the associated one-dimensional function and any two-dimensional interaction functions that include the variable of interest.

Once an ensemble of decision trees is trained using gradient boosting, all trees produced for a single predictor variable are used to predict the training samples and build the function associated with each feature. Once the trees are used to build the function for each predictor variable, they are no longer needed, simplifying inference to new data. Thus, the function associated with each predictor variable or interaction is derived from the large set of shallow trees as opposed to using a spline method, as is common for traditional GAMs.

Explainable Boosting Machines (EBM)



Maxwell, A.E., Sharma, M. and Donaldson, K.A., 2021. Explainable Boosting Machines for Slope Failure Spatial Predictive Modeling. *Remote Sensing*, *13*(24), p.4991.

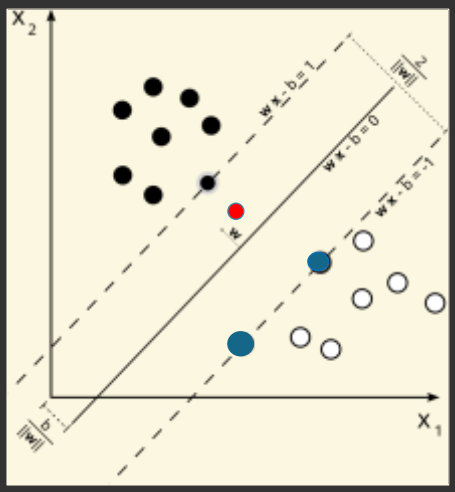
30

The figure provided on this slide conceptualizes the ancillary outputs of EBM that aid in interpretability. For the global model, results include (1) graphic output of the functions for each predictor variable and each included two-dimensional interaction and (2) an assessment of variable importance for each predictor variable and interaction term. For binary classification problems specifically, the predicted relationship between the predictor variable and the dependent variable is obtained by graphing the values of the predictor variable to the *x*-axis and the associated prediction or score to the *y*-axis. For included two-dimensional interactions, each variable will be mapped to an axis and the resulting prediction or score will be presented as a heat map within the two-dimensional space. As a result, components of the model can be represented graphically, which the algorithm originators cite as the key characteristic of an interpretable model. Larger scores indicate that the model associates those ranges of predictor variable values with a higher likelihood of occurrence of the positive class whereas lower values are associated with a lower likelihood or probability of occurrence.

Although EBMs are not widely used yet, I anticipate that they will see a wider adoption in the future due to their generally strong performance and interpretability. EBMs are a good example of how new algorithms and methods often build upon existing methods.

Support Vector Machines (SVM)

- Works with boundary conditions
- Find the linear boundary that gives the best separation between classes (hyperplane)
- Boundary defined by support vectors, or training samples that are nearest to the boundary
- Project the data to a higher dimension
- Two class problem
  - Combine multiple classifications to separate more than three classes
- Non-separable data
  - Positive slack variable (Cortes and Vapnik (1995))
  - Cost parameter (*C*)



http://en.wikipedia.org/wiki/Support_vector_machine

31

Support vector machines (SVMs) are a unique method that often shows strong predictive performance. RF and SVM are often my go-to methods.

The SVM algorithm attempts to define an optimal hyperplane to separate classes. The hyperplane is effectively the generated model. This hyperplane is the plane that provides the maximum margin or separation between classes and is defined by a subset of the available training samples, termed support vectors. In other words, only a subset of the available training samples are used to define the hyperplane or build the model.

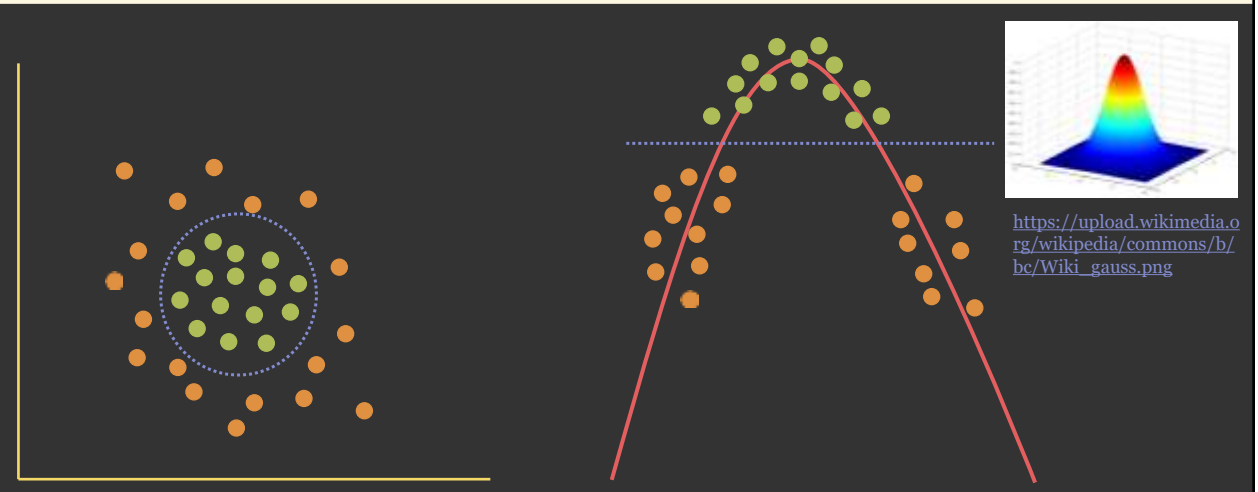
In order to allow for improved separation of classes that are not linearly separable, the training samples can be projected into a higher dimensional feature spaces using the “kernel trick” where the optimal hyperplane may be more linear. This process has greatly expanded the use of SVMs since we can now model relationships that are not linear.

The cost parameter is used to control the complexity of the generated hyperplane by controlling the penalty associated with misclassifying a sample, practically allowing for adjusting the model to compensate for overfitting or underfitting.

Methods have also allowed for the original SVM algorithm, which only allows

for the differentiation of two classes, to be expanded to allow for the differentiation of three or more classes.

The Kernel Trick



https://upload.wikimedia.org/wikipedia/commons/b/bc/Wiki_gauss.png

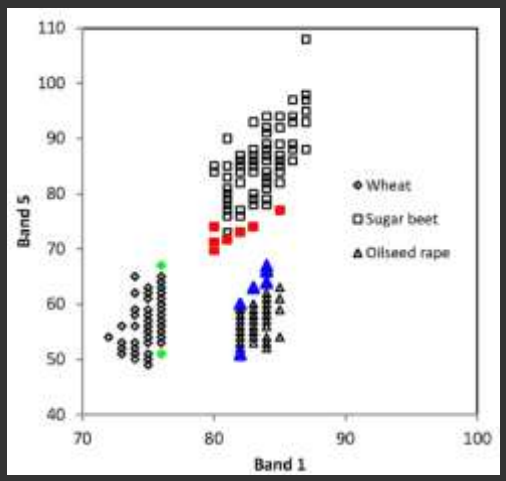
- Use the kernel trick to transform the feature space into a higher dimensional space where the features are more linearly separable

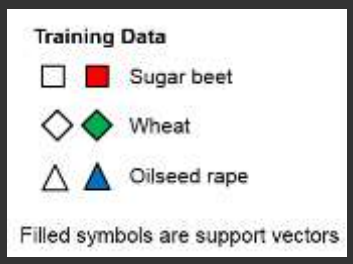
32

This slide further conceptualizes the kernel trick. Again, the goal is to project the input predictor variable space into a higher dimension in which a nonlinear boundary may become more linear and defined as a hyperplane.

As an example, imagine that the separating boundary between two classes in a two-dimensional space would best be defined as a circle, as pictured in the image on the left. The samples could be projected onto the surface of a three-dimensional curve where the separating boundary could be described as a plane. Thus, a nonlinear boundary was made more linear by projecting the data into a higher dimensional feature space.

Support Vectors



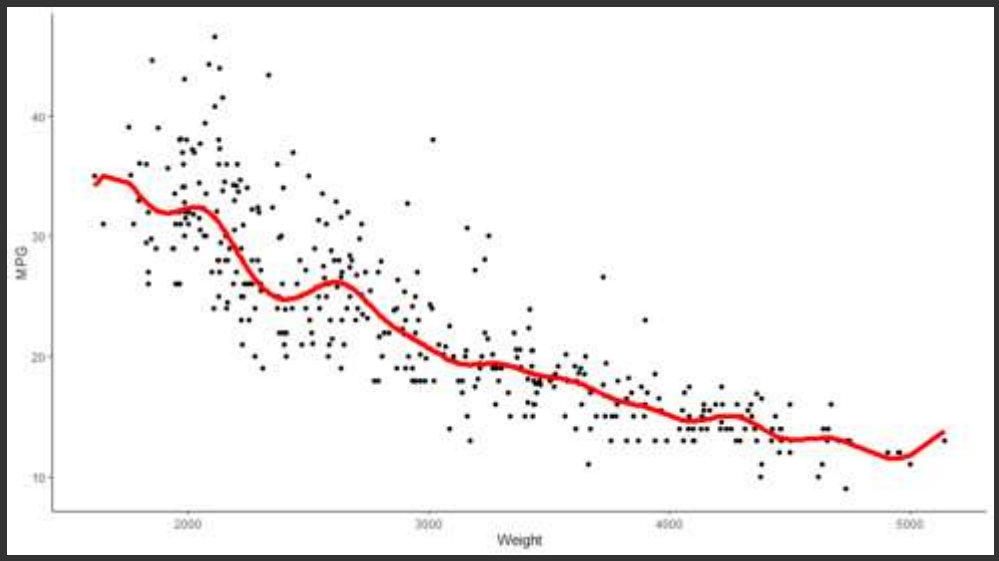


Pal, Mahesh, and Giles M. Foody, 2012. Evaluation of SVM, RVM and SMLR for accurate image classification with limited ground data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 5(5): 1344-1355.

33

This slide explains the concept of support vectors. In this example, three different types of plants are differentiated based on two variables. The colored-in dots represent the support vectors. These are the only samples used to define the hyperplane.

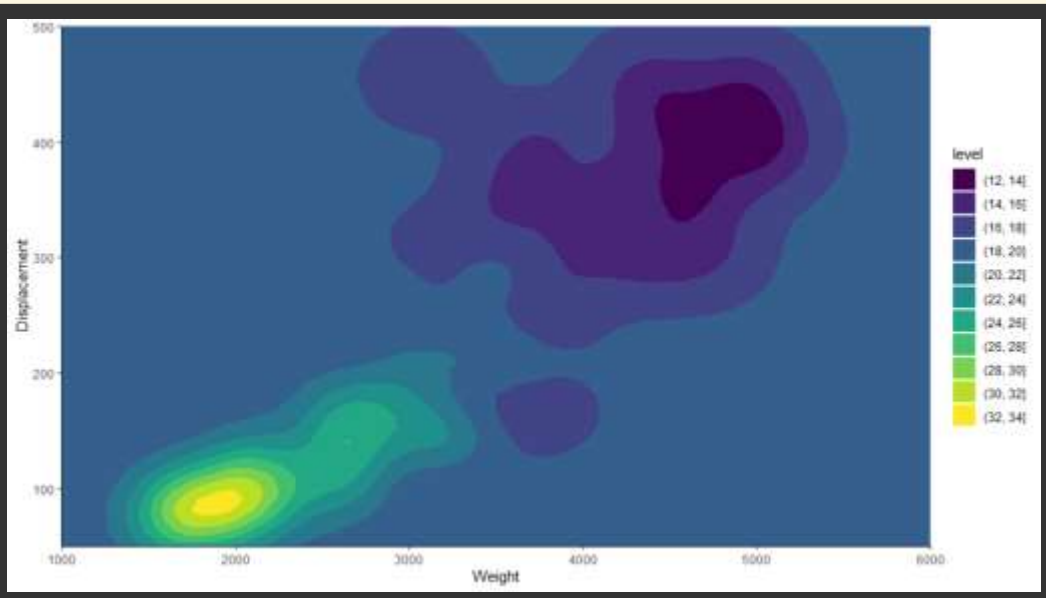
Example Data: Auto MPG



34

The red line on the graph represents the SVM model for predicting fuel efficiency using vehicle weight.

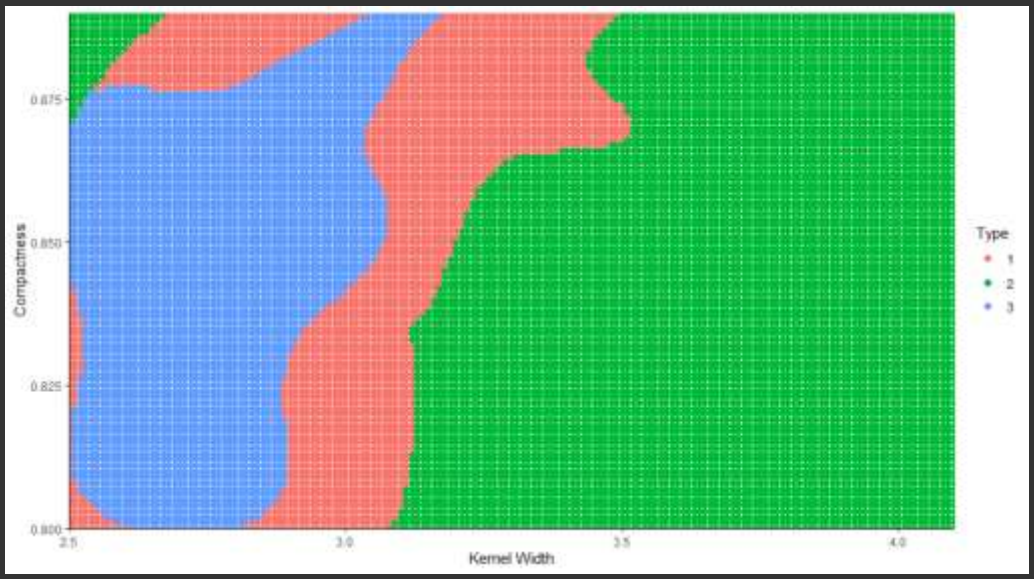
Example Data: Auto MPG



35

This two-dimensional surface represents the SVM model for predicting fuel efficiency using vehicle weight and engine displacement.

Example Data: Seeds



36

This result shows the decision space for separating the three varieties of wheat.

Support Vector Machines

Strengths

1. Nonparametric
2. Capable of regression and classification
3. No local minima issue
4. Not highly effected by outliers
5. Robust
6. Not sensitive to hyperparameters?

Weaknesses

1. Predictor variables must be on the same scale
2. Categorial predictors must be engineered
3. Sensitive to hyperparameters?
4. Can overfit

37

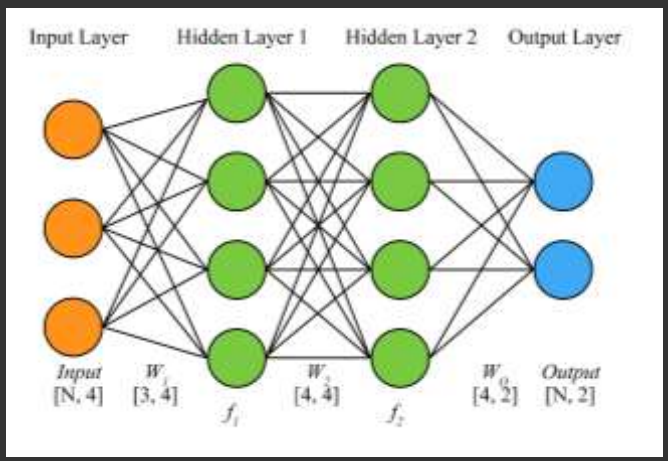
Strengths of SVM include that it is nonparametric, capable of performing regression and classification, does not suffer from the local minima issue (which will be discussed later in the context of artificial neural networks), is not highly affected by outliers, and is robust to complex datasets and relationships.

Some weaknesses are that all variables must be transformed to be on a similar scale. Further, categorical variables can only be included if they are converted into dummy variables. SVMs can also overfit, which is generally controlled by the cost or slack parameter.

Some have argued that SVMs are not sensitive to their hyperparameter settings. However, I have generally found that tuning the hyperparameters often improves model performance. As a result, I would suggest tuning the hyperparameters. These parameters include the cost parameter, which relates to the penalty applied for incorrectly predicting a training sample or a training sample falling on the wrong side of the decision boundary. A higher cost tends to result in overfitting to the training data whereas a lower cost may result in overgeneralization. The kernel used to transform the data to a higher dimensional space also has hyperparameters that can be tuned. For example, a polynomial kernel has an order parameter, and a radial basis function has a gamma parameter.

Artificial Neural Networks

- Consist of interconnected neurons or nodes
- Input layer represents predictor variables
- Output layer represents what is being predicted
- Model generated by learning a weight associated with each connection



38

Our final machine learning method example is artificial neural networks (ANNs). ANNs are often described as being modeled after the human brain, as they consist of interconnected neurons or nodes. However, how these learning algorithm functions are not necessarily comparable to the human brain, so I try to avoid using this analogy.

An ANN consists of neurons organized into layers as demonstrated on the slide. Input layers represent input predictor variables and have one neuron for each input, while output layers represent the desired output, such as the different categories for nominal data, and have one neuron for each output class.

Between the input and output layers are one or more hidden layers containing multiple nodes. Within the network, all neurons in a layer are connected to the neurons or nodes in adjacent layers, and these connections have weights.

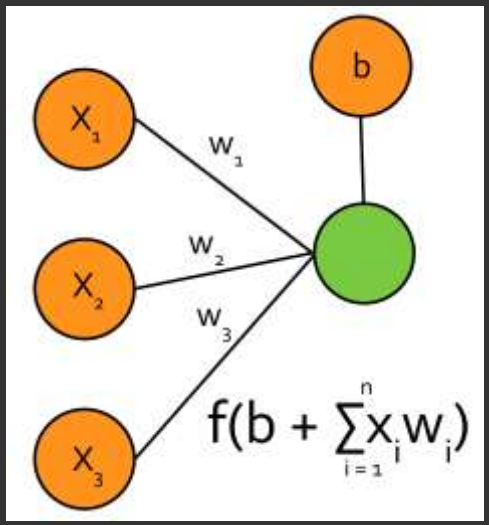
On the slide, the conceptualization contains three inputs or predictor variables and two hidden layers containing 4 nodes each. There are two classes being classified or predicted. Note that all nodes or neurons between adjacent layers are connected. These connections represent the weights.

Through the process of supervised learning, the weights in the network are

adjusted in an attempt to improve the prediction of the output.

Weights

- Weights are adjusted during training process
- Similar to slope or coefficient in regression
- Inputs from prior layer multiplied by weights
- Multiple Input X weight combinations are summed
- Weights relate to strength of activation or signal



39

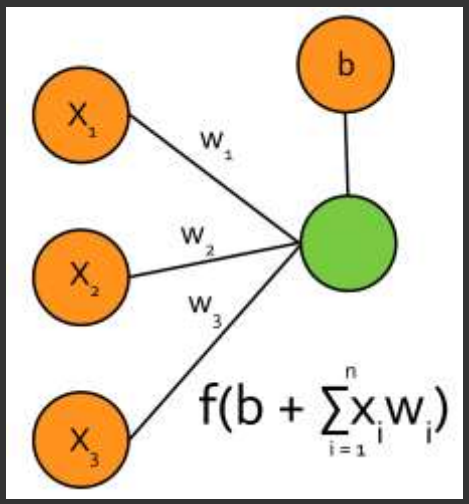
This slide further describes the concept of weights and how a signal or activation is produced at each node.

This is actually very similar to multiple linear regression. The weights are comparable to the coefficients for each predictor variable while the bias (b) is similar to the y-intercept.

So, at this point, the activation or output of the node is a linear combination of the inputs to the node, associated weights, and bias.

Bias

- A constant to shift the activation
- Similar to a y-intercept in linear regression



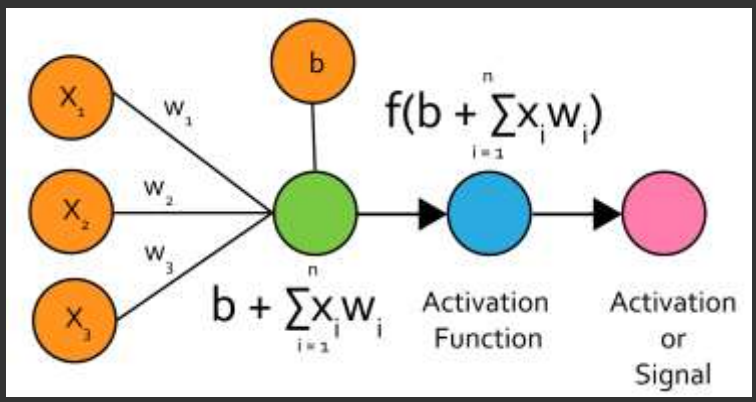
40

Again, the bias at each node is similar to the y-intercept in a linear regression equation. It is not multiplied by any input.

The bias allows the activation to be shifted to larger or smaller values, like an offset. The addition of a bias term further increases the flexibility of the model.

Activation Functions

- Allow for modeling of nonlinear relationships
- A mathematical gate between inputs and outputs



41

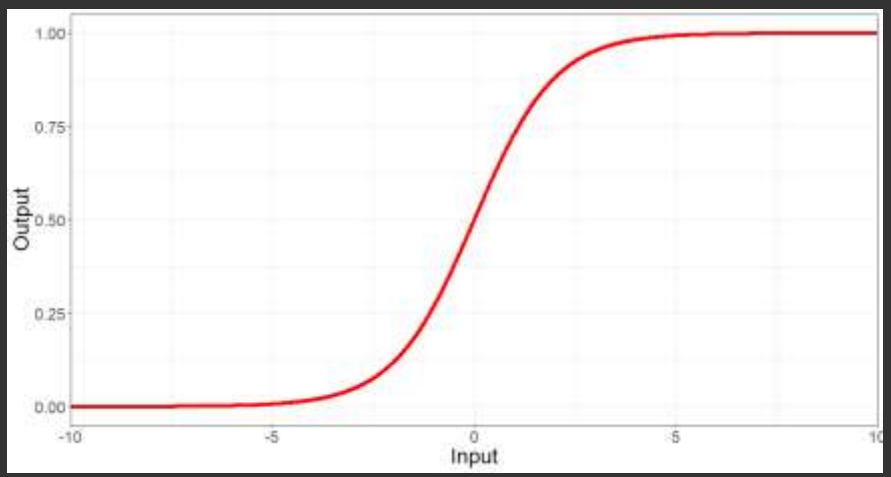
I left out an important component above in the discussion associated with weights, bias, and activation at each neuron or node. As I mentioned, this is similar to a multiple regression equation where the bias acts as a y-intercept, the weights act as coefficients, and the inputs act as the predictor variable values. In reality, this is more than similar to a multiple regression equation: it is a multiple regression equation.

In order to be able to model complex patterns and relationships between variables and make predictions from a complex and noisy signal, it would be useful to be able to model or describe non-linear, complex patterns in the data.

This is accomplished by (1) having many interconnected nodes, associated weights, and hidden layers and (2) applying an activation function to transform the activation or signal (or, the output from the neuron). Over the next few slides, I will provide some examples of activation functions.

Sigmoid or Logistic

1/(1+e^(-Input))

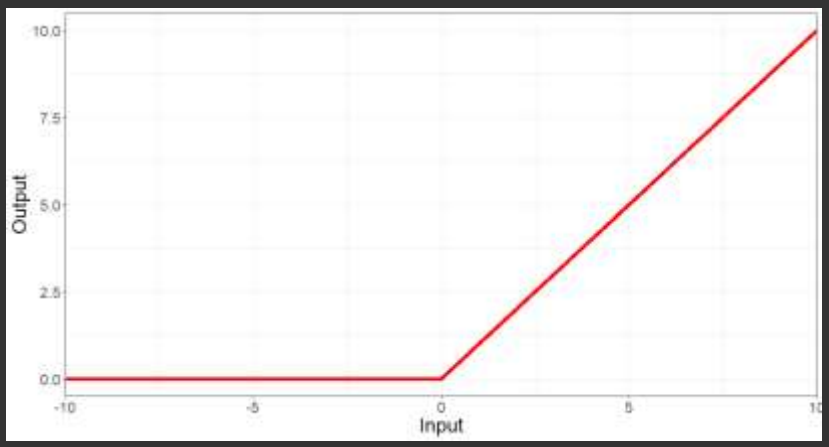


42

A sigmoid or logistic function is used to rescale the data from 0 to 1. This is generally used when the goal is to differentiate between two groups or categories and is similar to logistic regression.

Weights can be learned when using a sigmoid function as there is a gradient. However, there are some issues. First, the slope approaches zero for high or low activations. This can result in a vanishing gradient problem, where the gradient approaches 0 and the model becomes difficult to train. It is also not zero-centered because the mean activation is not near zero. In the example, it is near 0.5 since the output is scaled from 0 to 1. This can be an issue as it decreases strong positive and strong negative activation or signals.

Rectified Linear Unit (ReLU)

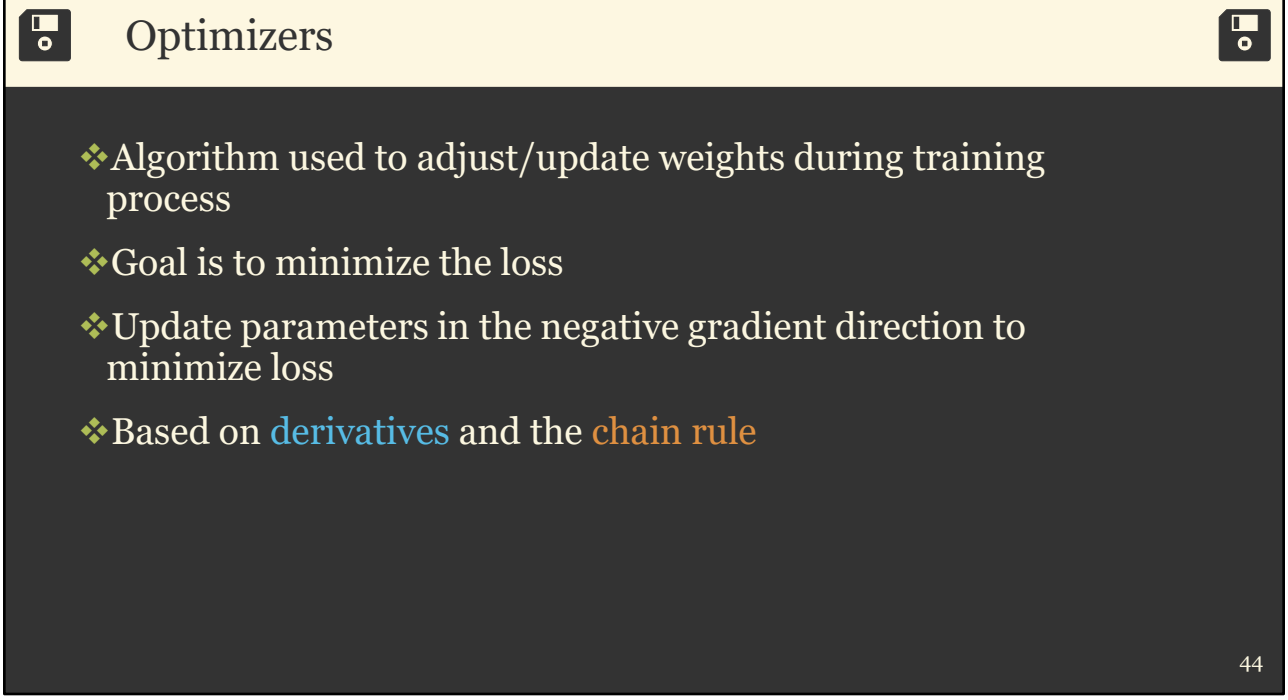


If Input > threshold:
    return Input
Else:
    return 0

43

The rectified linear unit (ReLU) activation function is currently commonly used. This allows for an activation of 0 below a certain threshold and a linear activation above a certain threshold.

This activation function does have some issues. First, it is not zero-centered and, second, since part of the function has a slope or gradient of 0, this can cause issues with backpropagation known as the “dying ReLU” problem. Despite some issues, it is still commonly used.



How are weights updated during the learning process? This is accomplished using an optimization algorithm.

The gradient descent algorithm is used to update the network parameters and weights to minimize the loss function during the process of supervised learning.

This minimum value is calculated using different algorithms, but the basic concept is gradient descent.

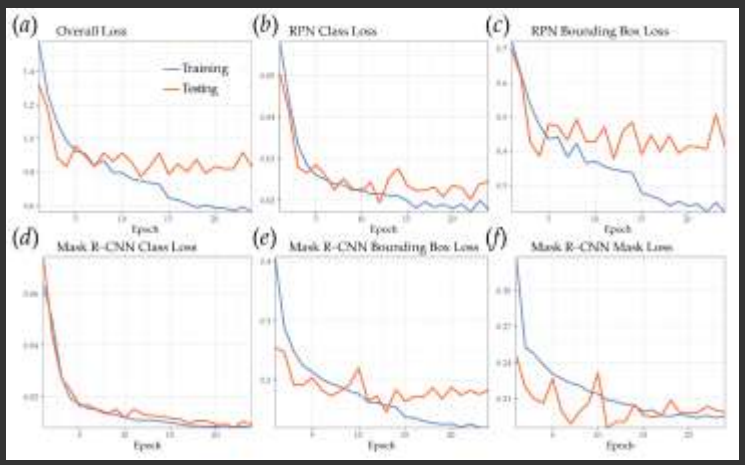
Derivatives and the chain rule are used to determine the gradient, or whether the model is moving toward the minimum or away from it. This knowledge can then be used to update the weights so that the model moves towards the minimum to decrease the loss. Since there are many weights that are calculated, the slopes, derivatives, or gradients are interrelated. So, this problem must be solved using the chain rule.

Although the mathematics of the optimization process are outside of the scope of this course, the important point here is that algorithms can be used to improve the loss over multiple iterations on the training set to gradually reach the optimal solution or minimum loss.

One issue is that the algorithm can get stuck in a local minima and fail to find or optimize to the best solution. Different methods are available to combat this issue.

Loss Functions

- Measure of error
- Used to assess learning during training
- Goal is to update weights to minimize the measure
- Use different measures for different situations
- Can created custom measures or use multiple measures



45

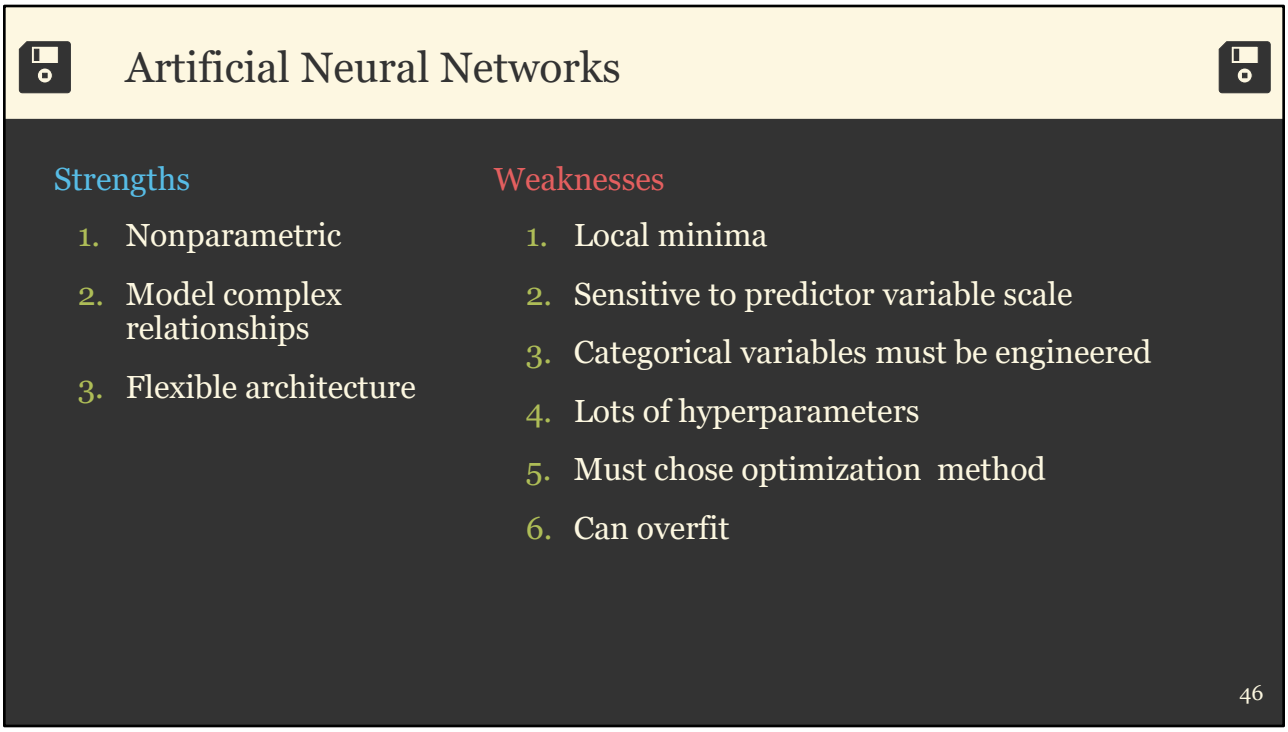
As the algorithm learns from the training data and updates the weights, it must monitor some measure of performance in order to know whether the predictions of the training data are improving and learning is occurring.

This is accomplished by monitoring a loss metric. Loss metrics are measures of error as opposed to measures of accuracy, and the goal is to reduce or minimize this metric via the learning processes.

A variety of loss metrics have been proposed, and you can even define a custom metric for a specific problem.

The appropriate loss metric partially depends on whether classification or regression is being performed.

During training, the backpropagation algorithm attempts to adjust the weights to minimize the loss metric. So, defining an appropriate loss metric is very important.

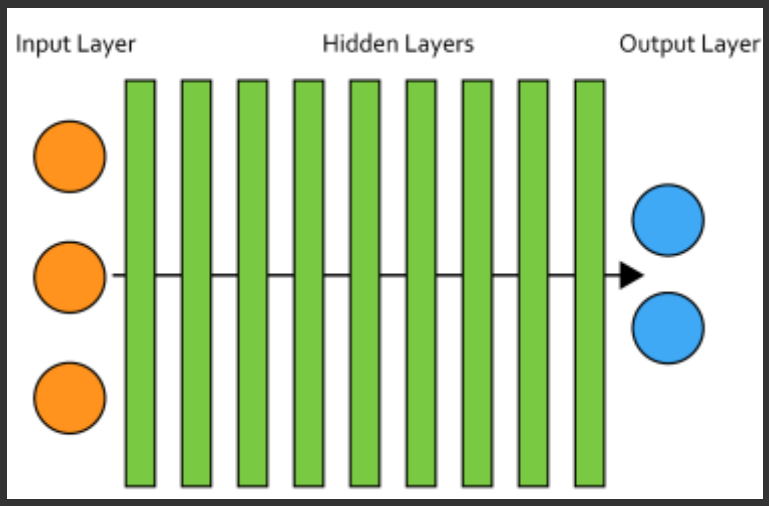


Artificial neural networks are nonparametric, can model complex relationships, and have flexible architectures for solving a variety of problems.

Issues with ANNs include not optimizing to the best solution (i.e., local minima), the need to rescale all predictor variables to a common scale and convert nominal predictors into dummy variables, a large set of hyperparameters to optimize, the need to choose an optimization algorithm or method to update the weights, and overfitting to the training data.

Deep ANNs vs ANNs

- More hidden layers
- Model more complex relationships



47

Deep learning, which has fueled many modern advancements in data science and predictive modeling, builds on ANNs and makes use of the flexible architectures they support.

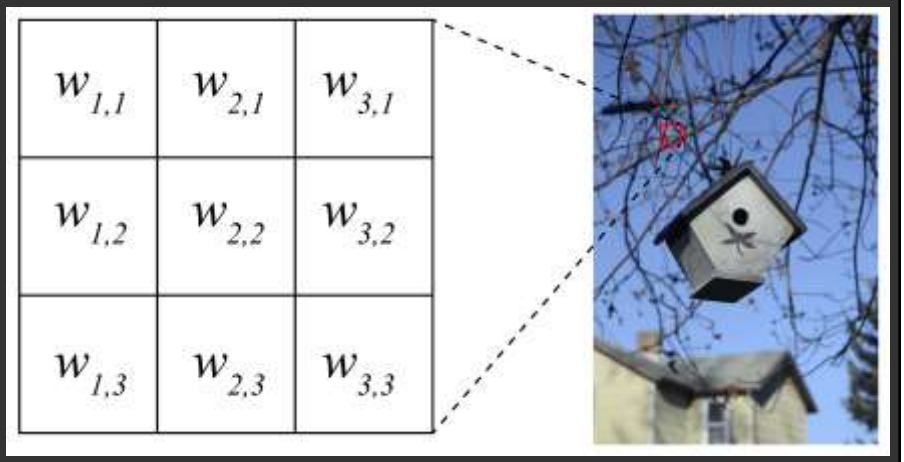
How is deep learning (DL) different from a traditional ANN? The key difference is the number of hidden layers, or layers between the input layer and output layer.

Deep learning algorithms generally have many hidden layers and associated nodes, as opposed to just one or two hidden layers.

So, you can think of "deep" as referring to the depth of the model or number of hidden layers. The idea is that more hidden layers allow for more data abstraction and improved model performance. However, the trade off is that there will be a larger number of weights to learn, which can increase the likelihood of overfitting, especially when the training set is small. Also, the complexity of the model results in longer training times and the potential need for more specialized computer hardware.

Convolutional Neural Networks

- CNNs or ConvNets
- Learn spatial patterns and context
- Based on moving windows
- Apply moving windows to create feature maps



48

Convolutional neural networks, ConvNets, or CNNs are a sub-type of DL ANNs that allow for the incorporation of spatial patterns into the learning process.

Instead of just learning weights associated with links between neurons, the ANN can also learn weights to build a moving window or kernel that is passed over the image to manipulate the data and learn spatial patterns and abstractions.

Once the filter or moving window is learned, it is used to alter the values in the array and produce a feature map.

The basic idea behind CNNs is to learn spatial patterns at different scales.

This type of DL has led to many of the recent improvements in computer vision and image analysis.

A full discussion of DL in general and CNNs in particular is outside the scope of this class. However, I wanted to at least mention these methods and how they relate to ANNs and more traditional ML methods.

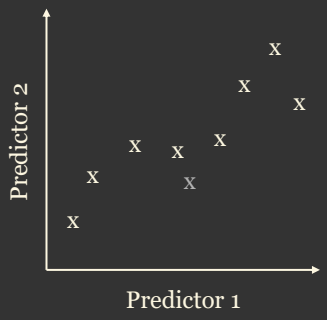
Unsupervised Learning

49

Before moving on, I want to introduce one example of an unsupervised classification algorithm. Remember that, in contrast to supervised learning, no training data are used to perform unsupervised learning. Instead, the algorithm clusters the data into groups based on common patterns and values in the associated predictor variables. So, unsupervised learning can be useful when training data are not available or to explore patterns in data.

k-Means

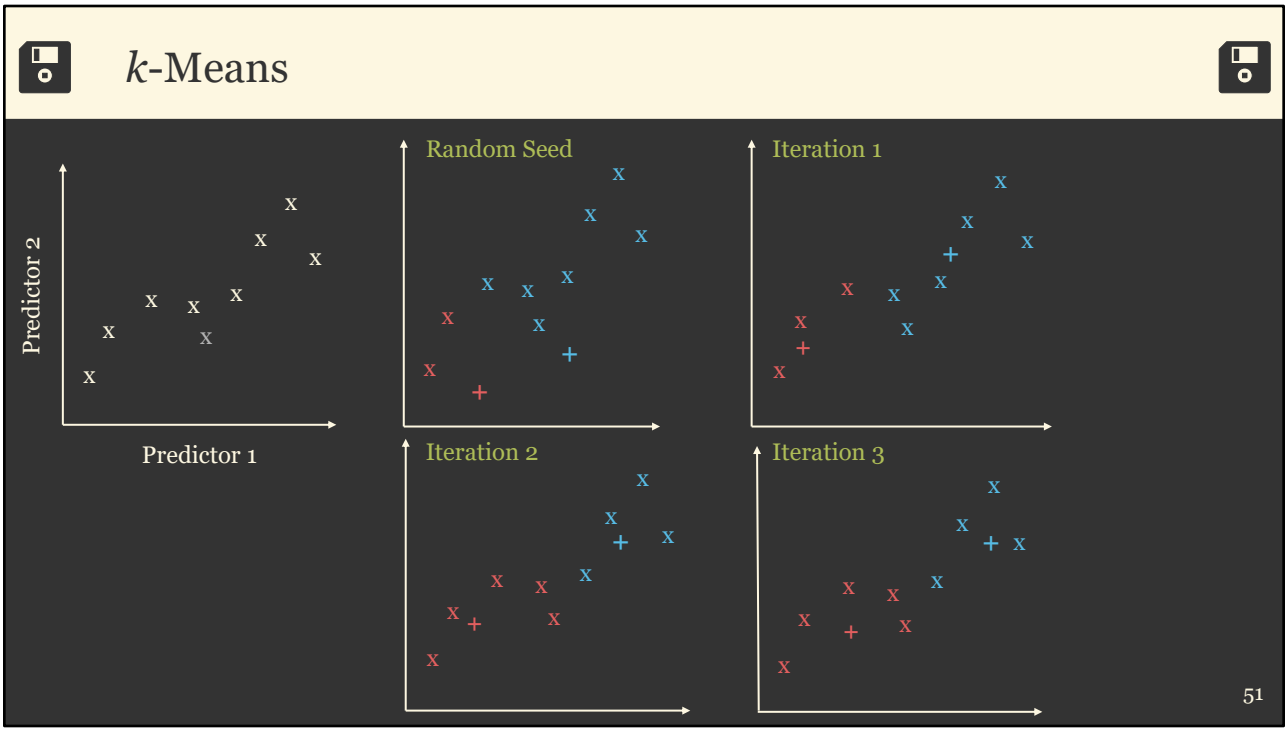
- Clustering method
- Algorithm arbitrarily "seeds" or locates the cluster centers in the multidimensional feature space
- Samples assigned to clusters
- Mean vectors for the clusters are determined
- Process is iterative
- Continues until mean vectors don't change significantly
- Augmentations of the method allows the number of clusters to be adjusted automatically during the iterations by merging similar clusters and splitting clusters with large standard deviations.



50

One commonly used algorithm for data clustering and unsupervised classification is *k*-means. To implement this clustering method, the analyst must define the number of desired clusters or groups. The algorithm then randomly seeds or locates cluster centers in the multidimensional measurement space. Data points are then assigned to these cluster centers based on minimal distance. This process continues through multiple iterations to move the cluster centers, or mean vectors, and reduce the overall distance between samples and these centers. The iteration process will stop once the centers no longer change significantly. Once the final mean vectors are obtained, all data points are assigned to clusters based on the mean vector they are closest to in the multidimensional space.

Since this method is based on distance, similar to *k*-NN, all the predictor variables must be on the same scale.



This slide graphically explains the *k*-means clustering process. To begin, the graph on the left shows the location of data points in a two-dimensional feature space defined by two predictor variables. To include more predictor variables, you can map to more dimensions. For example, with three predictors, you would map a location relative to three values to a location within the volume of a cube. It is difficult to visualize more than three spatial dimensions. However, locations can be mathematically described as a multidimensional vector. This allows for clustering to be performed in many dimensions. We will stick with two dimensions here for simplicity.

First, cluster seeds are randomly placed in the feature space. Samples are then assigned to these seeds based on proximity. In this example, the red Xs represent data points that have been assigned to the red plus sign mean vector while all the blue Xs have been assigned to the blue plus sign mean vector. In order to assess how well the mean vectors are positioned, the total distance of all the data points to their assigned mean vector are summed.

In subsequent iterations, the mean vectors are moved such that the distance is minimized between each mean vector and all points that were assigned to it in the prior iteration. The data points are then reassigned to cluster centers based on proximity. This process continues until a maximum number of iterations is reached, the mean vectors no longer need to be moved, or data points do not

need to be reassigned to lower the sum of the distances of each data point to its associated mean vector.

There are augmentations of *k*-means in which the number of clusters or mean vectors can change as the algorithm iterates over the data. Based on a minimum number of data points that can be mapped to a cluster, a maximum within-class variance to split clusters, and a minimum pairwise distance to merge clusters, mean vectors are updated, and clusters are merged and spilt.

Summary of Key Points: Machine Learning

- A variety of machine learning methods exist with different strengths and weaknesses.
- There is not an optimal algorithm for all tasks. It is generally best to assess the performance of multiple algorithms for specific problems.
- Modern advanced in deep neural networks and convolutional neural networks have resulted in significant recent advances in predictive modeling, especially in the field of computer vision.
- If training data are not available, unsupervised methods can be explored to search for natural clusters in data.

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

Thanks! Hope you found this useful.