Prof. Luong

Tyler Ferreira

Topics in Data Science

12/12/2022

Random Forest

Random forest is a type of ensemble method, which means that it combines the predictions of multiple individual models to make a final prediction. In the case of a random forest for image classification, the individual models are decision trees, and the final prediction is made by averaging the predictions of all the decision trees in the forest.

The accuracy given by the random forest classifier on the sign language dataset is: **81.54%**

AdaBoost

AdaBoost is another type of ensemble method, which means that it combines the predictions of multiple individual models to make a final prediction. In the case of AdaBoost for image classification, the individual models are decision trees, and the final prediction is made by weighting the predictions of the decision trees according to their accuracy on the training data.

The accuracy given by the adaboost classifier on the sign language dataset is: **31.55%**

Decision Tree

A decision tree works by creating a tree-like model of decisions, where each internal node in the tree represents a decision about the value of a feature, and each leaf node represents a class label. To make a prediction for a new image, the decision tree starts at the root of the tree and traverses the tree based on the values of the image's features, until it reaches a leaf node, which provides the predicted class label.

The accuracy given by the decision tree classifier on the sign language dataset is: **44.55%**

CNN

Convolutional neural networks (CNNs) are often used to model image data because they are able to automatically learn spatial hierarchies of features. This means that CNNs can learn to extract low-level features such as edges and corners from the raw input images, and then use these features to build higher-level features that represent more complex objects and patterns in the data. In the case of sign language data, a CNN can learn to extract features such as the shape of the hand and the position of the fingers in the images, and then use these features to classify the sign language gestures in the data. This is particularly useful because the spatial hierarchies of features learned by CNNs are able to capture the spatial relationships between different parts of the images, which are important for identifying and classifying sign language gestures.

It is necessary to transform the input data for a CNN model in order to make it more amenable to the network's structure and the learning process. This can help to improve the performance of the model and make it easier to train. For example, normalizing the images for CNN input helps to ensure that the model can learn from the data effectively. This is because the input data for a CNN should be in a standard format, with consistent ranges for pixel values. Normalization helps to make the data more amenable to this standard format, allowing the model to focus on learning to recognize the important features in the images rather than getting bogged down in the raw pixel values. Additionally, normalizing the data can help to improve the performance of the model by making it easier to train and reducing the risk of overfitting.

Furthermore, adjusting the learning rate is a valuable technique that can help to improve the performance of a deep learning model by ensuring that the learning rate is always set to an appropriate value. Using learning rate adjustment as a callback can help to make the training process more efficient and improve the performance of the model. It can also make the training process more robust, as the callback can automatically adjust the learning rate to account for changes in the data or the model architecture.

Additionally, data augmentation can help to reduce the risk of overfitting, which is a common problem in deep learning where the model performs well on the training data but poorly on new, unseen data. By generating additional data, we can make the training dataset

more diverse and reduce the risk of overfitting by forcing the model to learn more robust, generalizable features.

Here is the architecture details of the model:

1. The input layer consists of 28x28 grayscale images, each with a single channel (i.e. the grayscale value of each pixel).
2. The first layer is a convolutional layer with 32 filters and a 3x3 kernel. This layer applies 32 different 3x3 filters to the input images, producing 32 output images (or "feature maps") for each input image.
3. The second layer is a max pooling layer with a 2x2 pool size. This layer downsamples the output of the previous convolutional layer by taking the maximum value of each 2x2 region in each of the 32 feature maps. This reduces the spatial dimensions of the output from the previous layer by a factor of 2, while also reducing the number of parameters and computational cost of the model.
4. The third layer is another convolutional layer with 64 filters and a 3x3 kernel. This layer applies 64 different 3x3 filters to the output of the previous layer, producing 64 output feature maps for each input image.
5. The fourth layer is another max pooling layer with a 2x2 pool size. This layer further downsamples the output of the previous convolutional layer in the same way as the second max pooling layer.
6. The fifth layer is a third convolutional layer with 128 filters and a 3x3 kernel. This layer applies 128 different 3x3 filters to the output of the previous layer, producing 128 output feature maps for each input image.
7. The sixth layer is another max pooling layer with a 2x2 pool size. This layer further downsamples the output of the previous convolutional layer in the same way as the previous two max pooling layers.
8. The seventh layer is a flattening layer. This layer flattens the three-dimensional output of the previous convolutional layer into a one-dimensional vector, which can be fed into the dense layers of the model.
9. The eighth layer is a dense layer with 512 units and a ReLU activation function. This layer is a fully-connected layer, which means that all of the units in this layer are connected to all of the units in the previous layer. The ReLU activation function helps

to introduce non-linearity into the model, allowing it to learn more complex patterns in the data.

10. The final layer is the output layer, which has the same number of units as classes in the data. This layer uses a softmax activation function, which produces a probability distribution over the classes for each input image.

The accuracy given by the cnn model on the sign language dataset is: **98.5%***

*Bound to vary slightly due to variation in the value of the weights.

Performance Evaluation

Model Name	AdaBoost	Random Forest	Decision Tree	CNN
Accuracy	34.51%	44.55%	81.54%	98.5%

From the table above we can observe the difference in the performance of the traditional machine learning models and CNN.

Traditional machine learning models typically use hand-crafted features to represent the images, while CNNs learn these features automatically from the data. This means that CNNs can often achieve better performance than traditional models, since they can learn features that are more effective at representing the images.

Another difference is the way that the models are trained. Traditional machine learning models are trained by optimizing a loss function using gradient descent, while CNNs are trained using a variant of this method called backpropagation. This means that CNNs can often be trained more efficiently than traditional models, since backpropagation is a more efficient optimization method for deep learning models.

Overall, CNNs tend to be more effective than traditional machine learning models for image classification, due to their ability to learn effective features automatically and their efficient training algorithms. However, traditional machine learning models can still be useful in some cases, particularly when the amount of training data is limited or when interpretability is important.