Social Data Science

Supervised Machine Learning

Dr. Thomas Davidson

Rutgers University

November 3, 2021

Plan

1. Course updates
2. Classification algorithms
3. Intro to machine learning in R

Course updates

Homework 3

- Homework 3 due Monday at 8pm
- Project Prototype due Friday at 5pm
  - Requirements:
    - Link to a working ShinyApp on shinyapps.io
    - Link to a Github repository containing code for the app

Recap

- Supervised learning, unsupervised learning, and statistical inference
  - Supervised learning optimizes for predictive accuracy (focus on $\hat{Y}$ not $\hat{\beta}$)
- Over and underfitting
  - Out-of-sample validation and cross-validation
  - Regularization
- Evaluation etrics
  - Precision, recall, F1, ROC/AUC

Recap

- Given some outcome Y and a matrix of features X, we want to find a function $Y = f(X)$ that best predicts the outcome

Recap

Recap

Predicting penguins

- $Y = 1$ if the bird is a penguin, otherwise $Y = 0$
- X is a matrix including information on birds including their diet, wingspan, coloring, locations, etc.
  - Some of the information will be useful (e.g. ability to fly) but other information will be less meaningful (e.g. coloring)
- Goal is to find $f(X)$ to predict whether a given bird is a penguin
- The quality of the prediction will depend on both the information contained in X and the properties of the function $f()$.

Classification algorithms

Logistic regression

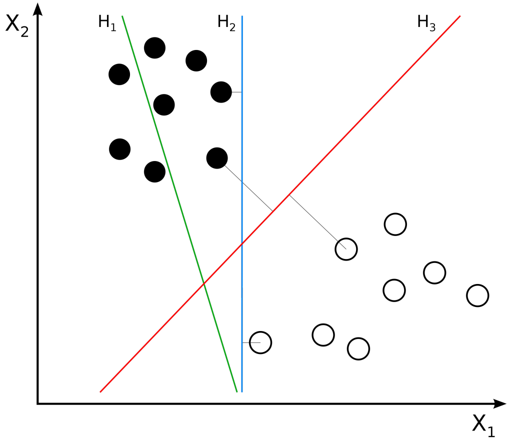
- Logistic regression is a regression model for the prediction of binary outcomes
- The model estimates the probability of an event ($Y = 1$) given predictors X.
- It uses a logistic function to ensure the output is a probability between 0 and 1.

$$P(Y_i = 1|X_i) = \frac{1}{1 + e^{-\beta_0 + \beta_1 x + \epsilon}}$$

- Multinomial logistic regression can be used if you have a multi-class outcome (more than two categories)
  - e.g. A model predicting level of education.
- A LASSO penalty can be used to regularize when many features are used.

Classification algorithms

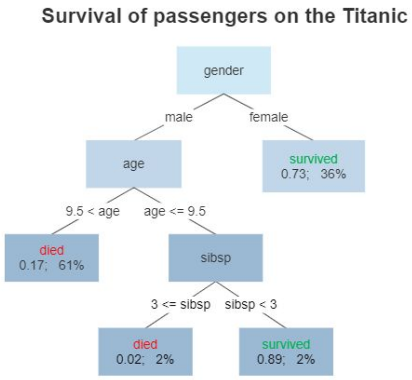
Support Vector Machines



Source: Wikipedia

Classification algorithms

Decision Trees



Source: Wikipedia. See this website for an excellent visual introduction to decision trees.

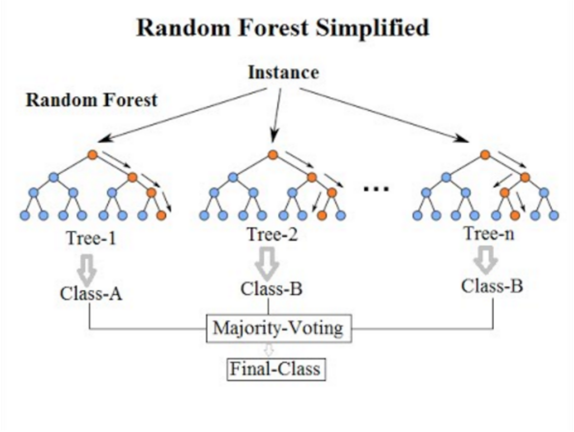
Classification algorithms

Random Forests

- Decision trees tend to overfit the training data
- Solution: Regularize by “growing” lots of trees and average over them
  - Using a procedure called *bootstrap aggregating*, or *“bagging”*, we can sample from our data and generate a *forest* consisting of many decision trees. - This is known as an *ensemble* method because it involves more than one model.
  - The approach is effective because the algorithm randomly splits the data into leaf nodes based on different features, hence it is a *random* forest.
  - The final classification is an average across the different decision trees.

Classification algorithms

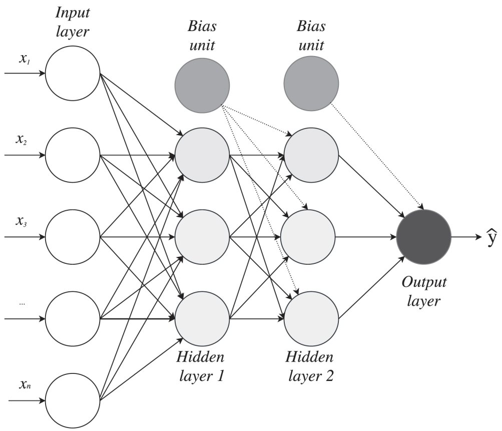
Random Forests



Source: Wikipedia

Classification algorithms

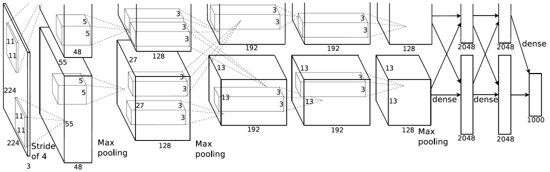
Neural networks



Davidson 2019.

Classification algorithms

Neural networks



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. "Imagenet Classification with Deep Convolutional Neural Networks." In Advances in Neural Information Processing Systems, 1097–1105.

Classification algorithms

Hyperparameters

- Each algorithm has hyperparameters that can adjust how it works.
  - e.g. Regularization type for logistic regression and SVM.
  - e.g. Number of trees, tree depth, and splitting criterion for random forest.
  - e.g. Number of layers, activation function, and optimization routine for neural networks.
- Often we conduct a search over different hyperparameters and compare many different models
  - This could also include different transformations of the feature matrix X
- But this can get very complex, very quickly!

Classification algorithms

Hyperparameter search and computational complexity.

- Davidson (2019) uses neural network models to predict high school GPA.
  - Three model hyperparameters with 40 different combinations
    - Number of hidden layers (depth)
    - Number of neurons per hidden layer (width)
    - Activation function ($f(X)$)
  - Each model is trained using 5-fold cross-validation, resulting in 200 different model fits
- These models took over 12 hours to estimate when run concurrently on a laptop

Python code and output is available here.

Classification algorithms

Black-box models and interpretability

- In contrast to standard explanatory models, which are considered to be interpretable, many of these algorithms are described as "black boxes," meaning that we are unable to observe their workings.
- There is a trade-off between model complexity (often associated with better predictions) and human interpretability
  - Watts (2014) argues that it may be worth sacrificing some interpretability in the interest of better predictions.
- But there are lots of developments in the field of ML interpretability
  - See Chrisoph Molar's open-source book *Interpretable Machine Learning for an overview.

Classification algorithms

Black-box models

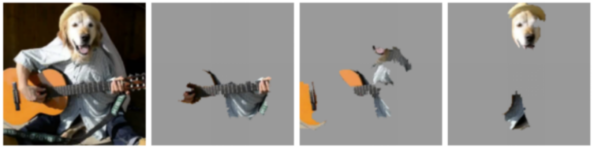
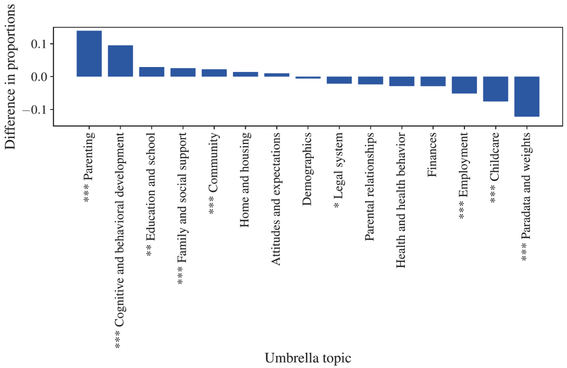

(a) Original Image
(b) Explaining *Electric guitar*
(c) Explaining *Acoustic guitar*
(d) Explaining *Labrador*

Figure 4: Explaining an image classification prediction made by Google's Inception network, highlighting positive pixels. The top 3 classes predicted are "Electric Guitar" ($p = 0.32$)**, "Acoustic guitar"** ($p = 0.24$) **and "Labrador"** ($p = 0.21$)

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. " 'Why Should I Trust You?': Explaining the Predictions of Any Classifier." In Proceedings of the 22nd ACM SIGKDD, 1135–44. ACM. https://doi.org/10.1145/2939672.2939778.

Classification algorithms

Black-box models

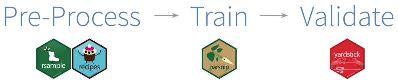


Davidson 2019.

Machine learning in R

Tidymodels

- `tidymodels` is a set of packages designed to use tidy principles to conduct machine-learning.
  - See https://www.tidymodels.org/packages/ for a list of packages.



Source: tidymodels tutorial.

Machine learning in R

Loading `tidymodels`

The `tidymodels` package loads all of the sub-packages, as well as the `tidyverse` packages. We're going to be using the `iris` dataset for today's analysis. This is a simple dataset containing data on 150 irises. There are three species, "setosa", "versicolor", and "viriginica" and four variables `sepal.Length`, `sepal.Width`, `Petal.Length`, and `Petal.Width`. The goal is to predict the species given the sepal and petal information.

```
library(tidymodels)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Machine learning in R

Loading and splitting data

We can use the `initial_split` command to create a train-test split, where 20% of the data are held-out for testing.

```
set.seed(12345)
iris_split <- initial_split(iris, prop = 0.8)
print(iris_split)
```

```
## <Analysis/Assess/Total>
## <120/30/150>
```

Machine learning in R

Viewing the data

```
iris_split %>% training() %>% head()
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 142          6.9         3.1          5.1         2.3  virginica
## 51           7.0         3.2          4.7         1.4 versicolor
## 58           4.9         2.4          3.3         1.0 versicolor
## 93           5.8         2.6          4.0         1.2 versicolor
## 75           6.4         2.9          4.3         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
```

Machine learning in R

Pre-processing

We will use the recipes package to pre-process the data.

```r
iris_recipe <- training(iris_split) %>%
  recipe(Species ~.) %>%
  step_corr(all_predictors()) %>%
  step_center(all_predictors(), -all_outcomes()) %>%
  step_scale(all_predictors(), -all_outcomes()) %>%
  prep()

iris_recipe # Note petal length removed due to correlation
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          4
##
```

Machine learning in R

Pre-processing the test data

The previous chunk only applied these transformations to the training data. We want to also modify the test data so that they are the same dimensions. We can apply the `recipe` to the new data using the `bake` command. We also want to load the training data into a variable using the `juice` command. This extracts the data directly from the recipe.

```
iris_testing <- iris_recipe %>%
  bake(testing(iris_split))

iris_training <- juice(iris_recipe)
```

Machine learning in R

Specifying a model

The parsnip command allows us to specify a model. ML models in R exist across a range of different packages and parsnip gives them a standardized syntax. We define the model, choose the package (in this case randomForest), then use fit to train the model.

```r
library(randomForest)
rf <-  rand_forest(trees = 1000, mode = "classification") %>%
  set_engine("randomForest") %>%
  fit(Species ~ ., data = iris_training)
```

Machine learning in R

Making predictions

```r
preds <- predict(rf, iris_testing)
bind_cols(iris_testing, preds)
```

```
## # A tibble: 30 x 5
##    Sepal.Length Sepal.Width Petal.Width Species .pred_class
##           <dbl>       <dbl>       <dbl> <fct>   <fct>
##  1       -0.195       1.75       -1.17  setosa  setosa
##  2       -0.913       1.51       -1.04  setosa  setosa
##  3       -0.913       0.552      -0.911 setosa  setosa
##  4       -1.03        0.791      -1.04  setosa  setosa
##  5       -0.793       2.46       -1.43  setosa  setosa
##  6       -1.75       -0.165      -1.30  setosa  setosa
##  7       -1.03        1.03       -1.17  setosa  setosa
##  8       -1.75        0.313      -1.30  setosa  setosa
##  9       -0.913       1.75       -1.04  setosa  setosa
## 10       -1.03        0.552      -1.30  setosa  setosa
## # ... with 20 more rows
```

Machine learning in R

Calculating metrics

```
precision <- bind_cols(iris_testing, preds) %>% precision(truth=Species
recall <- bind_cols(iris_testing, preds) %>% recall(truth=Species, esti
print(bind_rows(precision, recall))
```

```
## # A tibble: 2 x 3
##   .metric   .estimator .estimate
##   <chr>     <chr>          <dbl>
## 1 precision macro          0.944
## 2 recall    macro          0.933
```

Machine learning in R

Calculating metrics

We can also extract the predicted probabilities by adding an argument to the `predict` function.

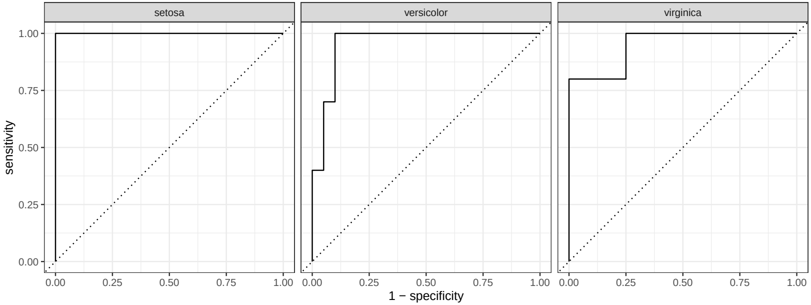
```
probs <- rf %>%
  predict(iris_testing, type = "prob") %>%
  bind_cols(iris_testing)
head(probs)
```

```
## # A tibble: 6 x 7
##   .pred_setosa .pred_versicolor .pred_virginica Sepal.Length Sepal.W
##          <dbl>            <dbl>           <dbl>        <dbl>       <
## 1        0.863            0.111           0.026       -0.195       1
## 2        0.999            0               0.001       -0.913       1
## 3        0.92             0.06            0.02        -0.913       0
## 4        1                0               0           -1.03        0
## 5        0.999            0.001           0           -0.793       2
## 6        0.988            0.003           0.009       -1.75       -0
## # ... with 2 more variables: Petal.Width <dbl>, Species <fct>
```

Machine learning in R

Calculating metrics

```
probs %>% roc_curve(Species, .pred_setosa:.pred_virginica ) %>% autoplo
```



Machine learning in R

Calculating metrics

```
probs %>% roc_auc(Species, .pred_setosa:.pred_virginica )
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.968
```

Machine learning in R

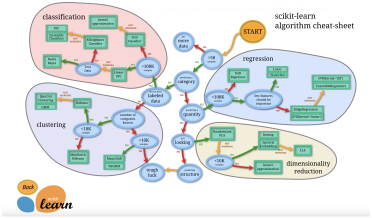
Next week

- We will go into more depth using `tidymodels` to implement cross-validation and a hyperparameter search and will evaluate multiple different models
- Supervised machine learning to perform text classification
- Considering how the inputs and label quality affect classifier performance

Machine learning in R

Alternatives

- Python has a more developed ML ecosystem than R.
  - `scikit-learn` provides a suite of tools for most machine-learning tasks except deep-learning, which requires specialized libraries.



Source: scikit-learn documentation. See this tutorial for how to run `scikit-learn` using R.