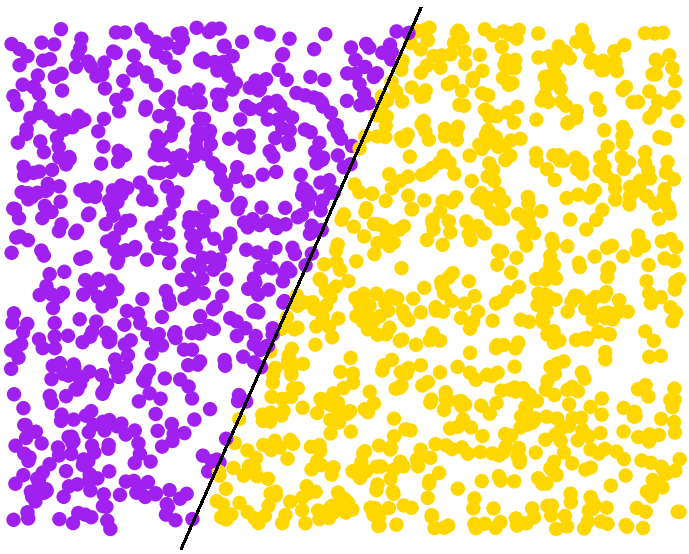


Figure 1: A linearly separable problem.

# Classifiers

### Logistic Regression

For much of the natural and social sciences, the goal of classification is inference. Inference of how much specific factors are associated with an observed phenomenon – not just prediction. The association typically are furnished with probabilistic qualities that allow an analyst to gauge how certain the pattern is. The output, in turn, lend themselves to building narratives.

The statistically-driven narrative are part of our daily lives. Nowadays, it would not surprise one to hear that a smoker has X-times higher chance of developing cancer than a non-smoker.[1] ^^MORE STATS EXAMPLES NEEDED HERE^^

These short empirical tid bits are rooted in a method known as *logistic regression*. Like ordinary least squares, logistic regressions are the workhorse of the social and natural sciences for inferring the marginal effects of input factors holding all else constant.

### Under The Hood

Before we dive into the particulars, we will cut straight to the chase: logistic regression is the workhorse of many fields, but it is truly well-suited for cases where we believe that the decision plane between two or more classes is a straight line. It imposes strong linear assumptions on a problem, which may not afford the flexibility of KNNs.

Let's assume that the classes of a target variable $y$ can be dinstinguished using some linear combination of input variables $x1$ and $x2$. Upon graphing the features and color coding using the labels, you see that the points are clustered such that purple points represent to $z = 1$ and gold points represent $z = 0$.

As it turns out, we can express the relationship between $y$, $x_1$, and $x_2$ as a linear model similar to OLS:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \epsilon$$

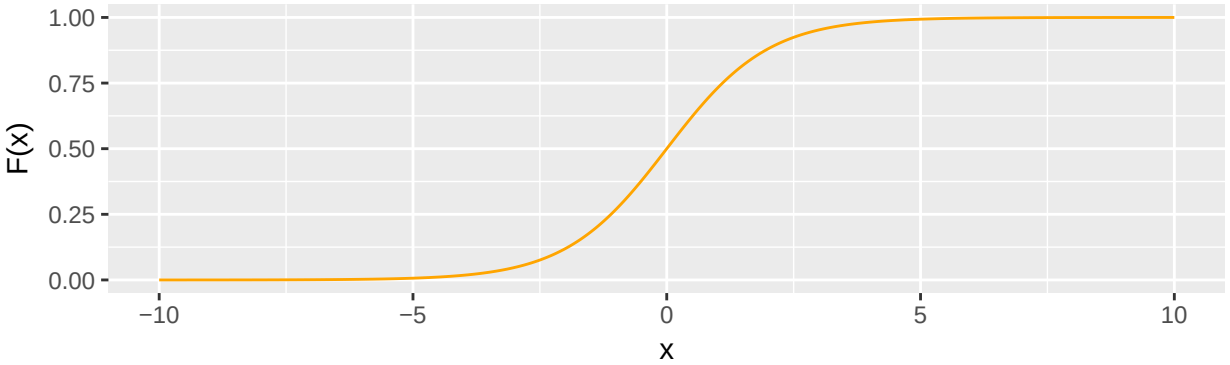
[1] https://www.cdc.gov/cancer/lung/basic_info/risk_factors.htm

where $y$ is a binary outcome and, like OLS, $\beta_k$ are coefficients that are learned using a *Maximum Likelihood Estimation* or MLE. The simple idea of MLE is that weights can be interatively adjusted so that we maximize the chance that the all the coefficients can jointly maximize the chance of accurately mimicking the target. While the innards of MLE are beyond the scope of this text, refer to *Elements of Statistical Learning* [2] or the more introductory version *Introduction to Statistical Learning.*[3]

If treated as a typical linear model with a continuous outcome variable, we run the risk that $\hat{y}$ would exceed the binary bounds of 0 and 1 and would thus make little sense. Imagine if $\hat{y}$, the predicted value of $y$ were -103 or +4 – *what would that mean in the case of a binary variable?* This is a logical shortcoming of a linear model for binary outcomes. Statisticians have cleverly solved the bounding problem by inserting the predicted output into a logistic function:

$$F(z) = \frac{1}{1 + e^{-z}}$$

For a variable $x$ that ranges for -10 to +10, the logit transformation converges to +1 where $x > 0$ and to 0 where $x < 0$. This S-shaped curve is known as a *sigmoid* and bounds $\hat{y}$ to a 0/1 range.




By substituting the linear model output $z$ into the logistic function, we bound the output between 0 and 1 and interpret the result as a conditional probability:

$$p = Pr(Y = 1|X) = F(z) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2)}}$$

This may seem to be a convoluted set of formulas, but it serves a convenient purpose. Unlike many of the techniques in this book, logistic regression is directly interpretable so as long as we believe that the decision boundary is linear. To be able to state that, for example, smokers have a 15 to 30-times higher chance of lung cancer than non-smokers, we can interpret coefficients as an odds ratio, implying that two quantities are compared. The odds of an event are defined as the following:

$$odds = \frac{p}{1 - p} = \frac{F(z)}{1 - F(z)} = e^z$$

In its purest form, probability $p$ can be calculated without a model, but to hold all covariates constant, we can fit the output of a logistic regression into this framework where $F(z)$ is a probability of some event $z = 1$and $1 - F(z)$ is the probability of $z = 0$. The odds can be re-formulated as:

$$pr(success) = \frac{e^{(w_0 + w_1 x_1 + w_2 x_2)}}{1 + e^{(w_0 + w_1 x_1 + w_2 x_2)}}$$

$$pr(failure) = \frac{1}{1 + e^{(w_0 + w_1 x_1 + w_2 x_2)}}$$

[2] Ref needed
[3] Ref needed

Typically, we deal with *odds* in terms of *log odds* as the exponentiation may be challenging to work with:

$$log(odds) = log(\frac{p}{1-p}) = w_0 + w_1x_1 + w_2x_2$$

where *log* is a natural logarithm transformation. This relationship is particularly important as it allows for conversion of probabilities into odds and vice versa.

The underlying coefficients of the logistic regression can be interpretted using *Odds Ratios* or *OR*. Odds ratios essential express a marginal unit comparison. Since $odds = e^z = e^{w_0+w_1x_1+w_2x_2}$, then we can express an odds ratio as a marginal 1 unit increase in $x_1$ comparing $odds(x+1)$ over $odd(x+0)$:

$$OR = \frac{e^{w_0+w_1(x_1+1)+w_2x_2}}{e^{w_0+w_1(x_1+0)+w_2x_2}} = e^{w_1}$$

After a little arithmetic, it turns out the OR is simply equal to $e^{w_1}$, which can be interpreted as a multiplicative effect or a percentage effect ($100 \times (1 - e^{w_1})\%$). More simply, this means that one can obtain the ballpark effect of a regression coefficient by exponentiating it. For example, if a logistic regression were trained to relate wages and citizenship binary variable ($x$) to whether people have health care insurance, the result may look as follows:

$$y(\text{no coverage}) = 0.468 - 0.048 \times wage + 0.372 \times \text{non-citizen}$$

As it, the coefficients provide little information other than the fact that a positive cofficient indicates that an increase in an input $x$ increases the chances of $y$. By exponentiating the cofficients odds of coverage are as follows for each variable:

- $OR_{wage} = e^{-0.048} = 0.953$ translates to -4.68% lower chance of not having health coverage for every $1000 increase in wages. Otherwise stated, more that one earns, better the chance of having health coverage.
- $OR_{non-citizen} = e^{0.372} = 0.451$ translates to 45% higher chance of not having health coverage among non-citizens.

**Tips**

Training a logistic regression is a fairly straight forward process as will be demonstrated in later sections. However, there are a number of key issues issues to keep in mind.

Tuning a logistic regression is a matter of selecting combinations of features (variables): finding the right combination of features will maximize classification accuracy. The process generally starts from a hypothesis of how variables are related to a target, but ultimately, the process unfolds as a series of trial and error tests. Suppose an analyst finds that a four-variable specification is the best model. If the underlying data set has 100 variables, then we can infer that the chosen specification is only one of 3.9 million possible four variable specifications. How does one know if the specification is the best? The possibilities of tuning an accurate model can be seemingly endless, but can be bound if the goals are set in advance. As logistic regression is one of the few truly interpretable machine learning algorithms, the goals can be broadly divided between *interpretability* (focus on $\hat{\beta}$) and *prediction* ($\hat{y}$).

**Interpretability**. Logistic regression is well-suited for socializing an empirical problem by crafting narratives around the coefficients. In order to tease out the associated effect of a variable, it is customary to estimate a series of models to understand how robust the observed effect size is.

- **Multicollinearity**. In virtually all introductory texts that cover logistic regression, collinearity is consistently flagged and rightfully so. As a refresher, multi-collinearity is a condition in which two or more input variables are not only correlated with the target variables, but amongst each other as well. The consequence is odd behavior among the coefficients – a factor that should be negative signals

positive, the size of effects may be extraordinarily large, etc. The answer lies within what the coefficients represent: they are the average effect of $x$ on $y$, partially isolated holding all else constant. Thus, if two or more variables have identical or very similar information, the model's learning process guided by MLE will be challenged in distilling the effects for each variable. This explains the odd behavior in coefficients and makes coefficients invalid for interpretation. Interestingly, the predictions $\hat{y}$ will still be usable. The best option is to conduct variable selection in advance to minimize double counting of signal.

- **"Ill-Posed Problems"**. Like ordinary least squares, logistic regression are not well suited for scenarios where the number of observations $n$ is out-numbered variables $k - k > n$. A more recent solution has been to rely on regularization methods (e.g. LASSO and Ridge) that are designed to efficiently perform in these scenarios.

**Sample imbalance**. ^^Sample imbalance will directly impact prediction accuracy^^

### Use Case: Health Care Coverage

Universal healthcare has become a basic human right in many countries. In the United States, this is not currently a guarantee, shrouded in heated political debate and controvery whether its a matter of human rights or a matter in which an individual may choose his or her fate. Regardless of the politics, there is data on healthcare coverage.

According to the American Community Survey ACS, an annual survey of approximately 3.5% of the US population as conducted by the US Census Bureau, over 22.4% of residents of the U.S. state of Georgia were without healthcare coverage in 2009. That is a fairly sizable proportion of the population – for every ten people, between two to three did not have coverage. If you read the news in 2010, a new law to provide affordable healthcare came into effect to help the uninsured.

Imagine that you are hypothetically tasked with getting the word out and drive recruitment in the state of Georgia. There is a hiccup, however. While commercial registries exist with people's demographic and personal contact information, most statistics on coverage are based on surveys, thus we do not precisely know *who* does not have insurance. A brute force approach could be to reach out to everyone under the sun though we can easily infer a wasted effort as 776 of every 1000 people are already covered. *How do we get to the 224 people who are not already insured?* For marketers, this is a classic targeting problem.

Data needs to enable the prediction and classification of a population into two classes: covered and not covered. By correctly classifying people as covered and not covered, decision makers and outreach staff can mobilize targeted outreach. From a data science perspective, the real objective is to be able to identify and replicate re-occurring patterns in the training data, then generalize the insights onto a sample or population that is not contained in the sample.

Given the label $y(Coverage)$, we can use logistic regression to not only infer what is associated with coverage, but also train a model to predict who is likely to need coverage:

$y(Coverage) = f(\text{Sex, Age, Education, Marital Status, Race, Citizenship})$

Based on the Census American Community Survey, we will illustrate how to construct a logistic regression. The sample has been *balanced*, meaning that both covered and non-covered survey respondents are represented in equal proportions in the sample.

```
# Load ACS health care data
  library(digIt)
  health <- digIt("acs_health")

# Convert characters into discrete factors
  factor_vars <- c("coverage", "mar", "cit", "esr", "schl")
  for(var in factor_vars){
    health[,var] <- as.factor(health[,var])
  }
```

```r
# Randomly assign
  set.seed(100)
  rand <- runif(nrow(health)) > 0.5

# Create train test sets
  train <- health[rand == T, ]
  test <- health[rand == F, ]
```

Training a logistic regression can be easily done using the `glm()` function, which is a flexible algorithm class known as Generalized Linear Models. Using this one method, multiple types of linear models can be estimated including ordinary least squares for continuous outcomes, logistic regression for binary outcomes and Poisson regression for count outcomes.

At a minimum, three parameters are required:

`glm(formula, data, family)`

where:

- `formula` is a formula object. This can take on a number of forms such as a symbolic description (e.g. $y = w_0 + w_1x_1 + w_2x_2 + \epsilon$ is represented as `y ~ x1 + x2`).
- `data` is a data frame containing the target and inputs.
- `family` indicates the probability distribution used in the model. Distributions typically used for GLMs are *binomial* (binary outcomes), *poisson* (count outcomes), *gaussian* (continuous outcomes - same as OLS), among others.

The family refers to the probability distribution family that underlies the specific estimation method. In the case of logistic regression, the probability family is *binomial*.

To start, we will specify three models:

- *Economic*: $coverage = f(log(age) + wage + employment)$
- *Social*: $coverage = f(citizenship + marital + schooling)$
- *Combined*: $coverage = f(log(age) + wage + employment + citizenship + marital + schooling)$

then assign each to a formula object and estimate each formula.

```r
# Formula objects
  econ <- as.formula("coverage ~ log(agep) + wage + esr")
  soc <- as.formula("coverage ~ cit + mar + schl")
  all <- as.formula("coverage ~ log(agep) + wage + schl + esr + cit + mar")

# Estimated GLM models
  glm_econ <- glm(econ, data = train, family = binomial)
  glm_soc <- glm(soc, data = train, family = binomial)
  glm_all <- glm(all, data = train, family = binomial)
```

In the social sciences and in public policy, the focus of regression modeling is typically placed on identifying an effect or an associated relationship that describes the process being studied. Often times, coefficient tables are examined, in particular the direction of the relationships (e.g. positive or negative weights), their statistical significance (e.g. p-value or t-statistics), and the relative fit of the model (e.g. the lowest Akaike Information Criterion or AIC provides *relative* model fit comparison). For example, an analyst may point out that education has an effect on coverage by interpreting the coefficient point estimates. In the combined model, education attainment coefficients are are estimated relative to people who hold a graduate degree, thus indicating that people who :

- did not finish high school have a *6.58-times* higher chance of not having health coverage ($ e^{w = 1.884} = 6.58$)

- hold a high school degree have a *4.91-times* higher chance of not having health coverage ($ e^{w = 1.592} = 4.91$)
- hold a college degree are relatively better off than the previous two groups with a *1.79-times* higher chance of not having health coverage ($ e^{w = 0.584} = 1.79$)

All coefficients are statistically significant. While it is valid to evaluate models on this basis, it is necessary to remember that this is not the same as evaluating a model for predictive use cases as predictive accuracy is not assessed on the basis of coefficients.

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Oct 19, 2018 - 23:37:00

Like the KNN example, the absolute accuracy of a model needs to be obtained through model validation techniques like cross validation. The `boot` library can be used to generate cross-validated accuracy estimates through the `cv.glm()` function:

```
cv.glm(data, glmfit, cost, K)
```

where:

- `data` is a data frame or matrix.
- `fit` is a glm model object.
- `cost` specifies the cost function for cross validation.
- `K` is the number of cross validation partitions.

Note that the cost function needs to take two vectors. The first is the observed responses and the second is the predicted responses. For example, the cost function could be the overall accuracy rate:

$$\frac{FP + FN}{TP + FP + TN + FN}$$

or the true positive rate (TPR):

$$\frac{TP}{TP + FN}$$

Both are written as functions below:

```r
# Misclassification Rate
  costAccuracy <- function(y, y.hat){
    a <- sum((y == 1 ) & (y.hat >= 0.5))
    b <- sum((y == 0 ) & (y.hat < 0.5))
    c <- ((a + b) / length(y))
    return(c)
  }

# True Positive Rate
  costTPR <- function(y, y.hat){
    a <- sum((y == 1 ) & (y.hat >= 0.5))
    b <- sum((y == 1 ) & (y.hat < 0.5))
    return((a) / (a + b))
  }
```

So that we can compare the cross validation accuracy with KNN, we will specify the `cost` using the misclassification rate for each of the three candidate models and set $k = 10$. Whereas KNN was able to achieve a 74% accuracy rate, the best GLM model was able to reach 72%, suggesting that some of the underlying variability in coverage rate is not captured in linear relationships. Also note that the input features for the KNN model were in a dummy matrix, thus the comparison is not perfect.

Table 1:

| | *Dependent variable:* | | |
|---|---|---|---|
| | coverage | | |
| | (1) | (2) | (3) |
| log(agep) | $-1.415^{***}$ | | $-0.785^{***}$ |
| | (0.045) | | (0.062) |
| wage | $-0.0005^{***}$ | | $-0.001^{***}$ |
| | (0.0001) | | (0.0001) |
| esrEmployed Civilian | $3.842^{***}$ | | $3.676^{***}$ |
| | (0.720) | | (0.723) |
| esrNot in Labor Force | $4.011^{***}$ | | $3.630^{***}$ |
| | (0.721) | | (0.724) |
| esrUnemployed | $5.261^{***}$ | | $4.937^{***}$ |
| | (0.723) | | (0.726) |
| citNon-citizen | | $2.132^{***}$ | $2.016^{***}$ |
| | | (0.083) | (0.084) |
| marMarried | | $-0.993^{***}$ | $-1.038^{***}$ |
| | | (0.059) | (0.061) |
| marNever Married | | $0.237^{***}$ | $-0.237^{***}$ |
| | | (0.060) | (0.071) |
| marSeparated | | $0.279^{**}$ | 0.207 |
| | | (0.125) | (0.130) |
| marWidowed | | $-1.837^{***}$ | $-1.459^{***}$ |
| | | (0.113) | (0.118) |
| schlHS Degree | | $1.712^{***}$ | $1.616^{***}$ |
| | | (0.102) | (0.103) |
| schlLess than HS | | $1.994^{***}$ | $1.937^{***}$ |
| | | (0.107) | (0.110) |
| schlUndergraduate Degree | | $0.648^{***}$ | $0.580^{***}$ |
| | | (0.114) | (0.115) |
| Constant | 1.166 | $-1.291^{***}$ | $-1.908^{**}$ |
| | (0.734) | (0.109) | (0.765) |
| Observations | 13,596 | 13,596 | 13,596 |
| Log Likelihood | −8,497.670 | −7,927.152 | −7,529.608 |
| Akaike Inf. Crit. | 17,007.340 | 15,872.300 | 15,087.220 |

*Note:* $^{*}$p<0.1; $^{**}$p<0.05; $^{***}$p<0.01

| specification | accuracy |
|---|---|
| Economic | 0.6482789 |
| Social | 0.6946161 |
| All | 0.7217564 |

In order to obtain the predicted values of *coverage*, we use `predict()`:

`predict(object, newdata, response)`

where:

- `object` is a GLM model object.
- `newdata` is a data frame. This can be the training data set or the test set with the same format and features as the training set.
- `response` indicates the type of value to be returned, whether it is the untransformed "link" or the probability "response".

We will now apply `predict()` to score the responses for each `train` and `test` samples.

```
  pred.glm.train <- predict(glm_all, train, type = "response")
  pred.glm.test <- predict(glm_all, test, type = "response")
```

A quick review of the predicted probabilities indicates confirms that we have the right response values (probabilities), bound by 0 and 1.

```
  summary(pred.glm.train)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## 0.0000006 0.3015294 0.5117123 0.4988232 0.6880606 0.9892398
```

Lastly, to calculate the prediction accuracy, we will once again rely on the combination of `ggplot2` and 'plotROC libraries for the AUC. Interestingly, the test set AUC is greater than that of the train set. This occurs occassionally and is often times due to the luck of the draw.

```
#plotROC
  library(plotROC)
  library(ggplot2)

#Set up ROC inputs
  input.glm <- rbind(data.frame(model = "train", d = train$coverage, m = pred.glm.train),
                     data.frame(model = "test", d = test$coverage,  m = pred.glm.test))

#Graph all three ROCs
  roc.glm <- ggplot(input.glm, aes(d = d, model = model, m = m, colour = model)) +
              geom_roc(show.legend = TRUE) + style_roc()  + ggtitle("ROC: GLM")

#AUC
  calc_auc(roc.glm)[,2:3]
```

```
##   group       AUC
## 1     1 0.7898052
## 2     2 0.7958594
```

**Practice Exercises**

1. Can logistic regression be applied to the downed tree problem from the KNN section? Apply the method to the downed trees data. How do the accuracies compare and why?

2. ^^Another question goes here^^