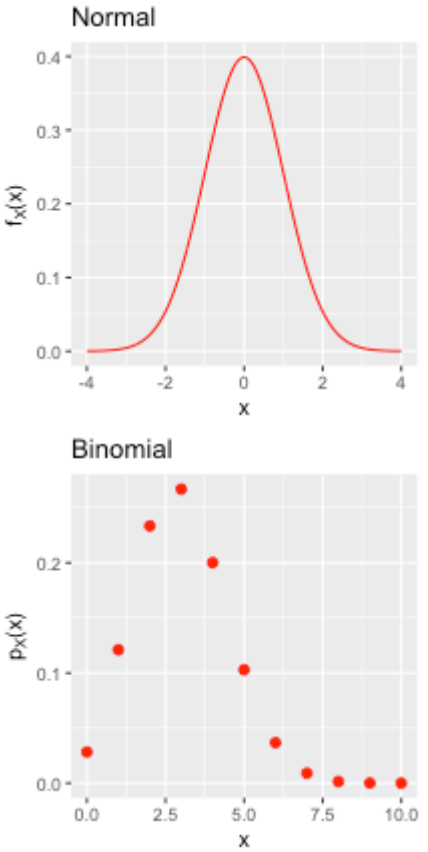
GLMs and Logistic Regression

36-600

Review: Probability Distributions

- A probability distribution is a mathematical function $f_X(x|\theta)$ where
  - x is discretely or continuously valued
  - θ is a set of one or more parameters governing the shape and location of the distribution (e.g., $\theta = \{\mu, \sigma^2\}$ for a normal)
- When x is discretely valued, $f_X(x|\theta)$ is a *probability mass function*, or *pmf*
- When x is continuously valued, $f_X(x|\theta)$ is a *probability density function* or *pdf*



Probability Distributions and Regression

- We are discussing distributions because in parameterized regression we make assumptions about how the response variable is distributed around the true regression line
- For instance, for simple linear regression, we assume that for every x...
  - the mean of the normal distribution is $\mu|x = E[Y|x] = \beta_0 + \beta_1 x$
  - the variance of the normal distribution is σ^2, which is a constant (i.e., does not vary with x)
  - the distribution governing the possible values of $Y|x$ has an infinite domain (and maybe it is a *normal* distribution)
- But...
  - what is $Y|x$ does not have an infinite domain?
  - what if $Y|x$ is not continuously valued?
- If one or both of these is the case, we would consider utilizing *generalized linear models*, or *GLMs*

Generalization: Choosing a Distribution

- In practice, there will be many possibilities and we might not know which one is right, but any assumption we make *should* be consistent with how the response is distributed
- Common assumptions that are made in practice:

Domain of Y	Consistent Distributions
$(-\infty, \infty)$	normal
$[0, \infty)$	Poisson, gamma, exponential, chi-square
$[0, n]$	multinomial
$[0, 1]$	beta, binomial

- Some of the distributions above are appropriate for continuous data (gamma, exponential, chi-square, beta) and some for discrete data (Poisson, multinomial, binomial)
    - foreshadowing: logistic regression assumes the binomial distribution

Generalization: Link Functions

- Let's assume we have one predictor, where the linear function is

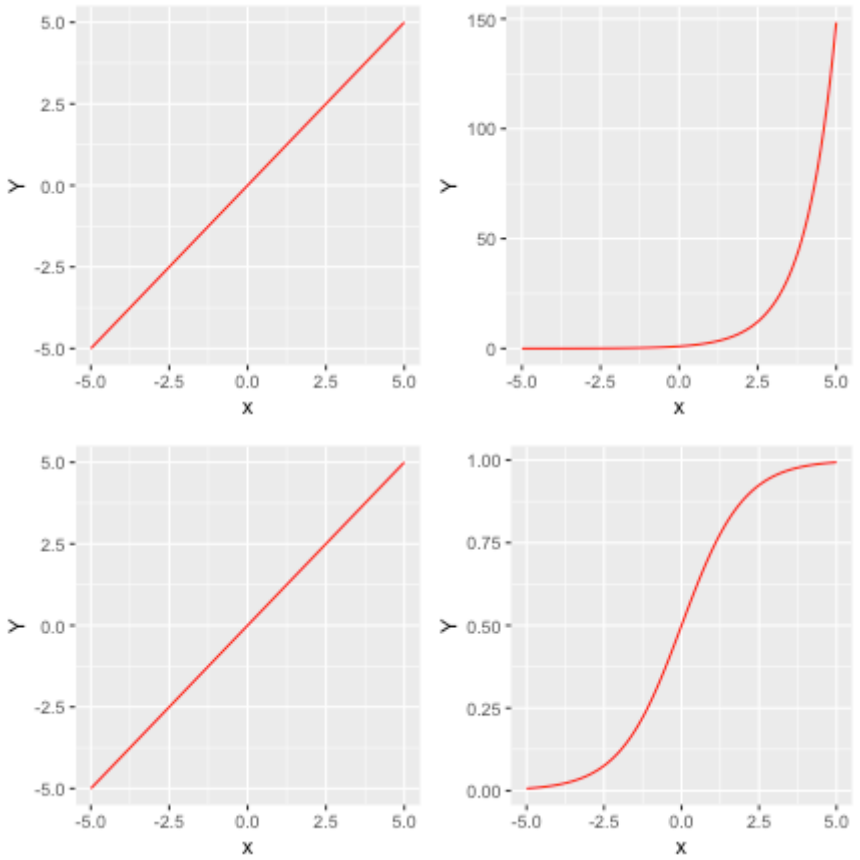
$$\beta_0 + \beta_1 x$$

- We use a *link function* $g(\cdot)$ to map this line to a restricted domain
- There are no unique transformations, but there are some that are commonly used
- To map to the domain $Y|x \geq 0$ (above right):

$$g(\mu|x) = \log(\mu|x) = \beta_0 + \beta_1 x \;\Rightarrow\; \mu|x = e^{\beta_0 + \beta_1 x}$$

- To map to the domain $Y|x \in [0, 1]$ (below right):

$$g(\mu|x) = \log\left(\frac{\mu|x}{1-\mu|x}\right) = \beta_0 + \beta_1 x \;\Rightarrow\; \mu|x = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$



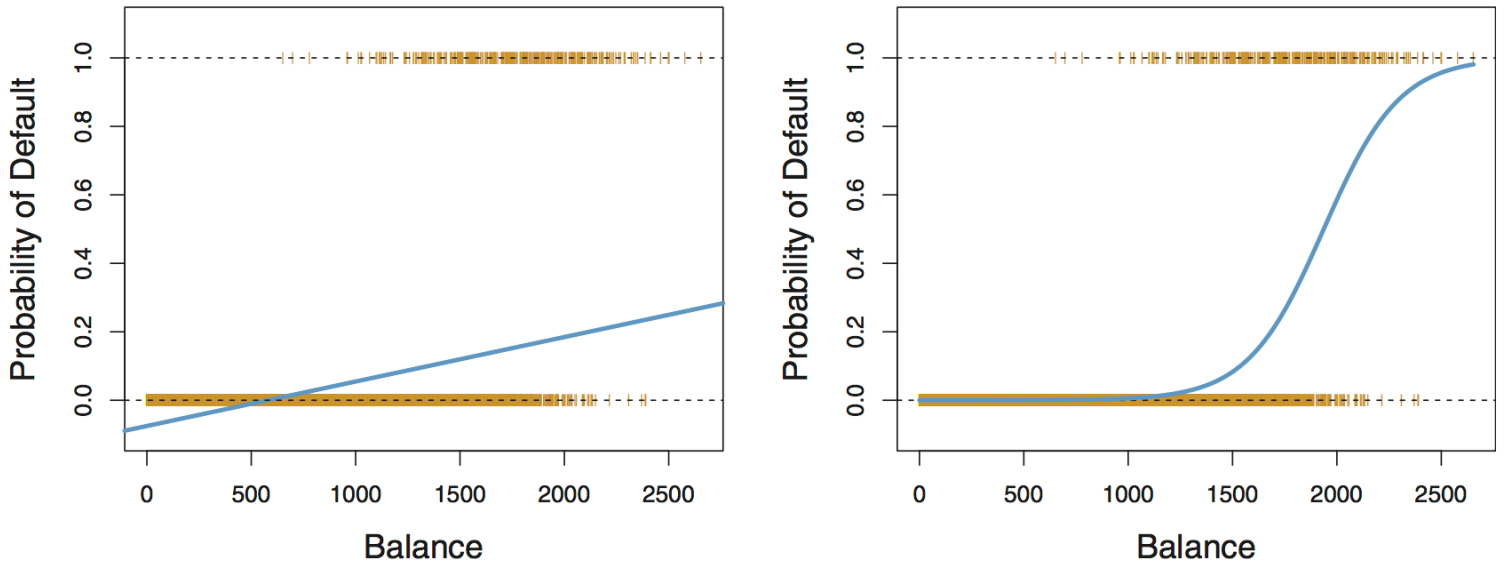
Generalization: Optimization

- Estimates β_0 and β_1 are made via maximization of the likelihood function, e.g.,

$$\mathcal{L} = \prod_{i=1}^{n_{\text{train}}} f_Y(y_i|\mu_i = e^{\beta_0+\beta_1 x_i})$$

  - how to read this: plug in guesses β_0 and β_1, determine μ_i given those guesses, evaluate the probability density function amplitude for Y_i given μ_i, and repeat for all data (and multiply the results together)
  - change β_0 and β_1 until this function attains its maximum value
  - the MLE is found via *numerical* optimization and thus GLMs are slower to learn than ordinary least squares models
- Of course, we need to *pick an appropriate distribution* (go back two slide) so that we have the functional form for $f_Y(y_i|\mu_i)$

Logistic Regression



(Figure 4.2, *Introduction to Statistical Learning* by James et al.)

- To the left is a linear regression fit. The regression line *is not* limited to lie within the domain [0,1]
- To the right is a logistic regression fit. The regression line *is* limited to lie within the range [0,1]

Logistic Regression

- Logistic regression is appropriate for datasets where the response variable takes on two discrete values (assumed to map to 0 and 1)
  - the underlying distribution is the *binomial distribution*, whose parameter is p, the probability of success (seeing outcome 1)
- Why is it named "logistic regression" and not "binomial regression"?
  - the conventional choice for the link function $g(p|x)$ is the *logit* function, seen on a previous slide (note: here, $\mu \to p$):

$$\log\left(\frac{p|x}{1-p|x}\right) = \beta_0 + \beta_1 x$$

- The probability of sampling a datum of class 1 at coordinate x_i is thus

$$p_i|x_i = \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}$$

- The likelihood function to be optimized is

$$\mathcal{L} = \left(\prod_{i:Y_i=1} p_i|x_i\right)\left(\prod_{i:Y_i=0}(1-p_i|x_i)\right)$$

  - we want $p_i|x_i$ to be large when $Y_i = 1$ and $p_i|x_i$ to be small when $Y_i = 0$

Logistic Regression: Inference

- In logistic regression, inference is done via *odds*
  - assume we have one predictor variable, and a datum with predicted response $p_i|x_i = 0.8$
  - that would mean that if we were to repeatedly sample response values at x_i, we would expect class 1 to be sampled four times as often as class 0:

$$O_i = \frac{p_i|x_i}{1 - p_i|x_i} = \frac{0.8}{1 - 0.8} = 4 = e^{\beta_0 + \beta_1 x_i}$$

  - the odds O are thus 4 (or 4-1 in favor of class 1)
- How do the odds change if we change the value of the predictor variable by one unit?

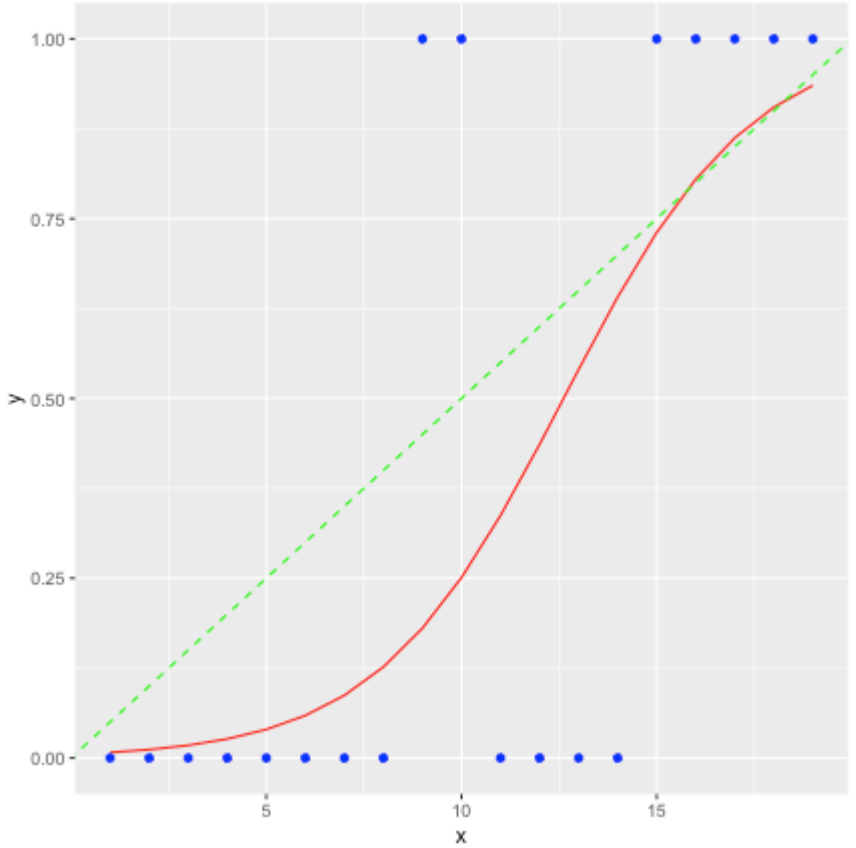
$$O_{\text{new}} = e^{\beta_0 + \beta_1 (x_i + 1)} = e^{\beta_0 + \beta_1 x_i} e^{\beta_1} = e^{\beta_1} O_{\text{old}}$$

  - if $\beta_1 > 0$, the odds go up as x_i increases
  - if $\beta_1 \gg 0$, the odds go up *quickly*; the sigmoid function $p|x$ quickly transitions from 0 to 1

Logistic Regression: Output

```
out.log <- glm(y~x,family=binomial)
ggplot(data=data.frame(x=x,y=out.log$fitted.values),
       mapping=aes(x=x,y=y)) + geom_line(color="red"
  geom_point(data=data.frame(x=x,y=y),
             mapping=aes(x=x,y=y),color="blue") +
  geom_abline(slope=0.05,intercept=0,
              color="green",linetype="dashed")
```

- The true $p|x$ is given by the green dashed line
- The estimated $p|x$ is given by the red line
    - note: a logistic function is "only so flexible" and may not replicate the truth



Logistic Regression: Output

```
summary(out.log)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.2800     2.3424  -2.254   0.0242 *
## x             0.4186     0.1843   2.271   0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25.008  on 18  degrees of freedom
## Residual deviance: 14.236  on 17  degrees of freedom
## AIC: 18.236
##
## Number of Fisher Scoring iterations: 5
```

```
logLik(out.log)   # the maximum log-likelihood value
```

```
## 'log Lik.' -7.117803 (df=2)
```

Logistic Regression: Output

- The model residual computed for each datum is the so-called *deviance residual*:

$$d_i = \text{sign}(y_i - \hat{p}_i)\sqrt{-2[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)]}$$

  - the sum of squares of the deviance residuals is $-2 \log \mathcal{L}$

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.2800     2.3424  -2.254   0.0242 *
x             0.4186     0.1843   2.271   0.0231 *
```

- the intercept is $e^{-5.28}/(1 + e^{-5.28}) = 0.005$
- the odds ratio is $O_{\text{new}}/O_{\text{old}} = e^{0.4186}$

Logistic Regression: Output

```
    Null deviance: 25.008  on 18  degrees of freedom
Residual deviance: 14.236  on 17  degrees of freedom
AIC: 18.236
...
'log Lik.' -7.117803 (df=2)
```

- The maximum value of the log-likelihood function is -7.118
  - the sum-of-squares of the residual deviances is -2 times -7.118, or 14.236
  - the AIC is $2k - 2\log\mathcal{L} = 2 \cdot 2 - 2 \cdot (-7.118)$ = 18.236, where k is the number of degrees of freedom (here, `df` = 2)
  - note that these metrics do *not* tell you whether the model represents the data-generating process well in an absolute sense (see the Hosmer-Lemeshow test for further details)
- Recall: when selecting models, select the one with the lowest AIC

Logistic Regression: Predictions

- In the previous example, there was no training/testing split
- In "real" analyses, there would be...you'd learn the model and generate test-set predictions by running the following code

```
resp.prob <- predict(out.log,newdata=pred.test,type="response")
resp.pred <- rep(NA,length(resp.prob))
for ( ii in 1:length(resp.prob) ) {
  if (resp.prob[ii] > 0.5) {
    resp.pred[ii] <- "<class 1>"    # FILL IN THE NAME OF CLASS 1
  } else {
    resp.pred[ii] <- "<class 0>"    # FILL IN THE NAME OF CLASS 0
  }
}
```

- `resp.prob` is a number between 0 and 1: if that number is less than 0.5, we predict that the test datum is associated with class 0, otherwise we predict it is associated with class 1
- In a future lecture, we will re-examine our use of 0.5 as a threshold for class splitting

Model Diagnostics: Classification

- The most straightforward diagnostic tool for assessing a classification model is the *confusion matrix*
  - the rows are predicted classes
  - the columns are observed classes
- To create a confusion matrix:

```
resp.prob = predict(out.log,newdata=pred.test,type="response")   # same as on the last slide
resp.pred = ifelse(resp.prob>0.5,"<class 1>","<class 0>")        # compressed if-else
mean(resp.pred!=resp.test)                                       # compressed MCR calculator
table(resp.pred,resp.test)                                       # confusion matrix
```

Model Diagnostics: Classification

- Here's an example of a confusion matrix:

	class.test	
class.pred	QSO	STAR
QSO	129	39
STAR	28	104

- There are *many* metrics associated with confusion matrices
  - the *misclassification rate*, or *MCR*, is the ratio of the sum of the off-diagonal values in the confusion matrix (top right and bottom left) to the overall table sum (0.223 above)
  - *accuracy* is simply 1 — MCR
  - see, e.g., this web page so as to be overwhelmed by all the possible choices