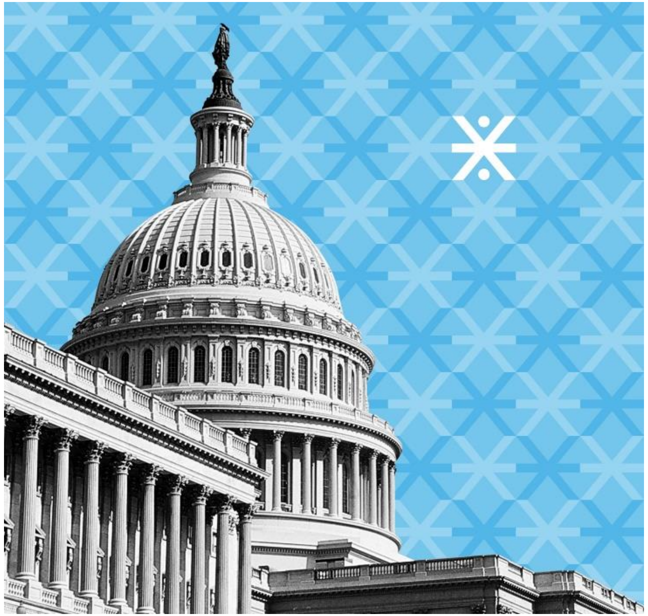
summit

complexity simplified.

Generalized Linear Models

Tim Book

Summit Consulting, LLC



Summit.describe()

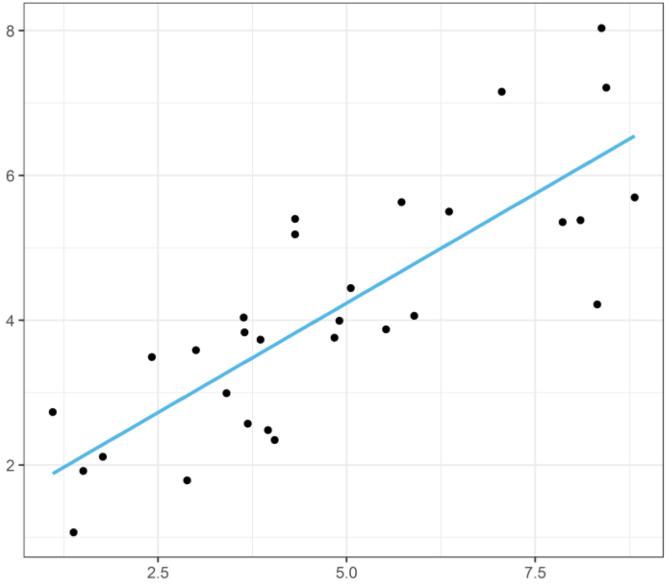
- Summit is a statistical and econometric consulting firm here in DC.
- Work with both government agencies and private companies.
- I work in our Litigation directorate. I use data science to help lawyers win cases.
- Summit is currently hiring. (www.summitllc.us)

Table of Contents

“Recap” of OLS

OLS At-A-Glance

- **Ordinary Least Squares is the classic "line of best fit."**
- **Given:**
  - x-variable(s).
  - y-variable on a ***continuous scale***.
- **We find:**
  - The slope estimate $\hat{\beta}$. (How do we interpret this again?)
  - The variance estimate σ^2.
- **But what do we mean by *best* fit?**



OLS At-A-Glance

- **The traditional OLS has the following form:**

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi} + \varepsilon_i$$

For each $i = 1, \ldots, n$

- **We also assume** $\varepsilon_i \sim \text{iid } N(0, \sigma^2)$.
- **So now we can think of each** $y_i \sim \text{ind N}\left(\mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta}, \sigma^2\right)$.
- **Notice in this notation I eliminated the need for thinking about residuals.**

Finding $\widehat{\boldsymbol{\beta}}$

- **Now, if each $y_i \sim N(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}, \sigma^2)$, we can use *maximum likelihood estimation* to find the "best" $\widehat{\boldsymbol{\beta}}$:**

$$\hat{\beta} = \operatorname*{argmax}_{\beta} f(\mathbf{y}|\boldsymbol{\beta}) = \operatorname*{argmax}_{\beta} \log f(\mathbf{y}|\boldsymbol{\beta})$$

$$= \operatorname*{argmax}_{\beta} \left(c - \sum_{i=1}^{n} \frac{1}{2\sigma} \left(y_i - \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} \right)^2 \right)$$

$$= \operatorname*{argmin}_{\beta} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} \right)^2$$

Now THAT looks familiar!

So…

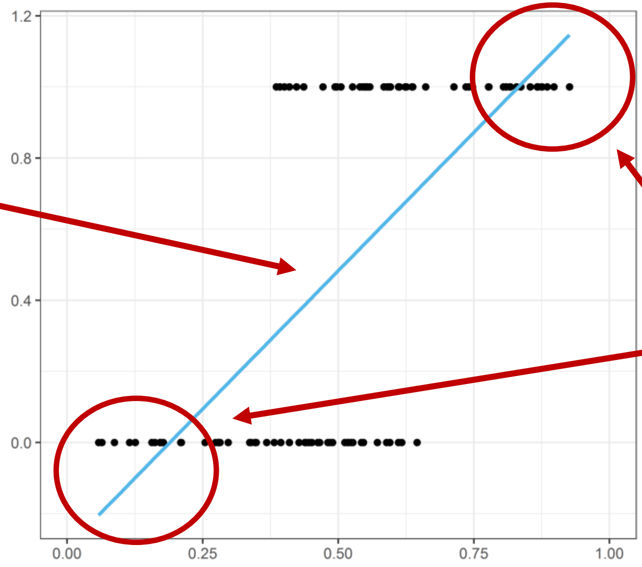
- **So the "least squares" technique is actually just a maximum likelihood technique!**
- **The maximum likelihood technique relied heavily on our** ***normality assumption*****.**
- **But what if our response variable y isn't normal? What if it's not even** ***continuous*****?!**

“Recap” Logistic Regression

Binary Response

- **Let's first consider the times where our response variable y is binary (coded as 0 or 1).**
- **This can be any type of variable where there are *exactly two* possible outcomes: yes/no, male/female, republican/democrat, on/off, etc.**

- **What would happen if you tried to fit an OLS model to a binary response?**



What do values in the middle mean?!

What's going on here?!

What's the problem?

Besides the mechanical issues, using OLS to model binary response suffers two major flaws:

1. **The range of responses is *unbounded*.**

2. **A binary random variable is not *normally distributed*.**

Solution

Given data $\mathbf{x}$ (I leave it out of what follows for brevity), we can assume our binary response follows a Bernoulli distribution with success probability p:

$$y \sim \text{ind Ber}(p)$$

Notice that we can't actually ***observe*** p! Can only observe y.

Nonetheless, let's try to get this to look like a linear model anyway. We need a function that maps p (which can only be in $(0,1)$) to the real line $\mathbb{R}$.

What values can this have?

$$p$$

What values can this have?

$$\frac{p}{1-p}$$

And does this value have a name?

What values can this have?

$$\log \frac{p}{1-p}$$

This is also sometimes written $\operatorname{logit} p$

Finally, a linear model!

$$y_i = \log \frac{p_i}{1 - p_i} = \mathbf{x}_i^{\mathrm{T}} \beta$$

Notes:

- **Remember that p, like y, depends on $\mathbf{x}$**
- **I didn't add the '$+\ \varepsilon_i$'. Why not?**

Rearranging...

Since

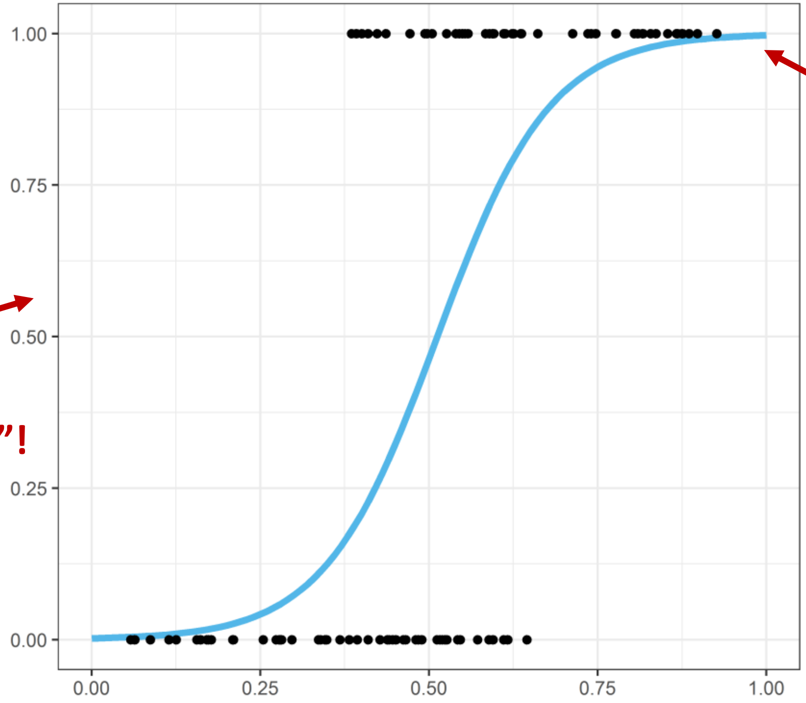
$$y_i = \operatorname{logit} p_i = \log \frac{p_i}{1 - p_i} \in \mathbb{R}$$

We can rewrite as

$$p_i = \frac{e^{y_i}}{1 + e^{y_i}} = \frac{e^{\mathbf{x}_i^{\mathrm{T}} \beta}}{1 + e^{\mathbf{x}_i^{\mathrm{T}} \beta}} \in (0,1)$$

(This is sometimes called the "inverse logit" function.)

Does it work?



Bounded!

Thanks to Bernoulli assumption, y-axis can now be interpreted as the "*p*-axis"!

Solving…

So, uh, how do we now find the βs? Since we know y is Bernoulli, let's try that ***MLE*** technique again…

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} f(\boldsymbol{y}|\beta) = \underset{\beta}{\operatorname{argmax}} \log f(\boldsymbol{y}|\beta)$$

$$= \underset{\beta}{\operatorname{argmax}} \log\left(\prod p^{y_i}(1-p)^{1-y_i}\right)$$

$$= \cdots \text{substituting and rearranging} \cdots$$

$$= \underset{\beta}{\operatorname{argmax}} \sum y_i\left(\mathbf{x}_i^{\mathrm{T}}\beta\right) - \sum \log\left(1 - e^{\mathbf{x}_i^{\mathrm{T}}\beta}\right)$$

Gross and impossible!

IRLS

For logistic regression (and all (most) other (basic) GLMs), the coefficients β are determined via maximum likelihood and *Iteratively Reweighted Least Squares* (*IRLS*).

Hand-wavy reminder of what IRLS is:

IRLS = Newton's Method + Chain Rule

Interpretation

- **OLS:** *"For a one unit increase in x_j, we would expect a $\hat{\beta}_j$ increase in y, holding all other variables constant."*

- **Logistic Reg:** 🤔

Reminder: $\text{logit}(p) = \mathbf{x}^{\mathrm{T}}\hat{\beta}$

Reminder: $\frac{p}{1-p} = \mathrm{e}^{\mathbf{x}^{\mathrm{T}}\hat{\beta}}$

Interpretation

- **OLS:** *"For a one unit increase in x_j, we would expect a $\hat{\beta}_j$ increase in y, holding all other variables constant."*

- **Logistic Reg:** *"For a one unit increase in x_j, we would expect the odds of [success] to* ***increase by a factor*** *of $e^{\hat{\beta}_j}$"*
- **Why?**

$$\frac{p}{1-p} = e^{\hat{\beta}_0+\hat{\beta}_1(x+1)} = e^{\hat{\beta}_0+\hat{\beta}_1 x} e^{\hat{\beta}_1}$$

GLMs In General

GLMs have 3 components

- ***Linear (Systematic) component:*** The *x*-variables and corresponding slopes.
- ***Random component:*** The distributional assumption you make about where the "randomness" in your model comes from. (eg, the Bernoulli distribution in logistic regression)
- ***Link component:*** The mapping function that "links" the random component to the linear component. (eg, logit)

Linear (Systematic) Component

- **This is the part you already know about. I won't spend a lot of time here.**

$$stuff = \underbrace{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}_{\textit{This}}$$

Random Component

- **This is where you make you distributional assumption about your response.**
  - Earlier, we assumed our 0/1 response was ***Bernoulli***. In general, we could have assumed our response was ***binomial***, if that made sense to us.
  - In OLS, we assumed our response was ***normal***. (Yes, OLS is really just a GLM!)

What else is out there?

- **Actually, any distribution that is a member of the *Aitken Exponential Family* can be used in a GLM!**

$$c(y,\phi)\exp\left\{\frac{b(\theta)T(y)-A(\theta)}{a(\phi)}\right\}$$

Look ugly? It is. Here are some Cliffsnote:

Commonly used Aitken EFs:

- Normal
- Bernoulli/Binomial
- Poisson
- Exponential/Gamma
- Multinomial (much more on this later!)

Less common ones:

- Inverse Normal
- Negative Binomial

Link Component

- **The link component links the random component to the linear component. It must be invertible and (usually) fit this criterion:**

$$g\colon \mathcal{Y} \longrightarrow \mathbb{R}$$

Response space (eg, (0,1) for logistic reg.)

The range of $\mathbf{x}^{\mathrm{T}}\beta$

Example: Logistic Regression Link

- **The most common link function for logistic regression is the *logit function*, which we saw earlier:**

$$\mathrm{logit}(p) = \log \frac{p}{1-p}$$

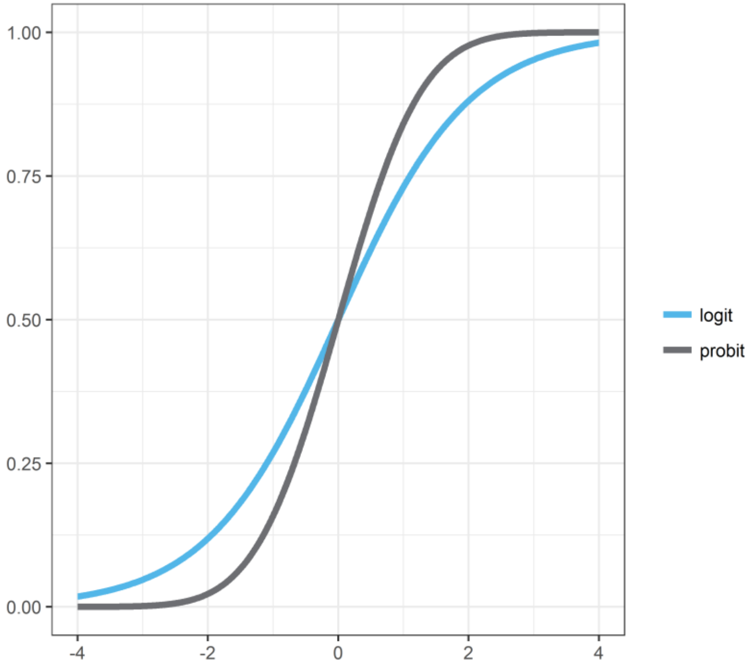
- **Another common one is the *probit function*:**

$$\mathrm{probit}(p) = \Phi^{-1}(p)$$

- NOTE: If you use the probit function, it is ***NOT*** called "probistic regression" (citation: Ellie Kazar)

They both look the same, so why choose?

- Visually, the logit and probit links are very hard to tell apart. Choosing one versus the other probably won't affect your predictions very much.
- The difference, then, comes in ***interpretation***, if you care about that.
- The logit allows you to discuss your probabilities in terms of ***odds***.
- The probit allows you to discuss your probabilities in terms of ***percentiles***.



Formalizing the Logistic Regression

In purest mathematical terms:

$$Y_i \sim \text{ind Bin}(1, p_i)$$

$$\text{logit}(p_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}$$

Other Common GLMs

Poisson Regression

- ***Poisson regression*** **is very popular for modeling** ***count data*****, especially when the counts are small.**
- **For example: Modeling household size versus number of bedrooms, number bathrooms, etc**
  - Household sizes start at 1 person, most being around 4 or 5, but can go higher.
- **Poisson regression typically uses the** ***log link*****.**
  - Why? Counts are typically very right-skew. The natural log can pull unlikely large numbers lower, making the overall distribution appear more linear.
- **You can also use the** ***square root link*****, but it is less common.**
  - It is, however, the variance stabilizing link!

Formally, Poisson Regression:

$$Y_i \sim \text{ind Poi}(\mu_i)$$

$$\log(\mu_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}$$

Discuss: How is this different from just taking the log of the response in a usual regression?

Gamma Regression

- **Although it is rarer than the others, *gamma regression* can be used when your response is *waiting times* between events in a *Poisson process*.**
  - Think: Length of time for a lightbulb to go out

- **There are two competing links for gamma regression: the *log link* and the *inverse link*. How do you decide?**
  - If the plot of y vs log(x) looks linear, use the log link
  - If the plot of y vs 1/x looks linear, use the inverse link.

Formally, Gamma Regression:

$$Y_i \sim \text{ind Gamma}(\mu_i, \nu)$$

$$1/\mu_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Where $\phi = 1/\nu$ is the *dispersion parameter*, which is used for model inference (more on this in a minute).

The Boring Stuff (Model Inference)

Slope Inference

- **Just like in OLS, the slope parameters $\hat{\beta}$ obey the Central Limit Theorem.**

- **So, as $n \longrightarrow \infty$,** $\hat{\beta}_j \longrightarrow N\left(\hat{\beta}_j, SE(\hat{\beta}_j)\right)$

Slope Inference

Y tho? We often want to remove variables that aren't "important" to prevent overfitting and multicollinearity. We then test:

$H_0: \beta_j = 0$

$H_A: \beta_j \neq 0$

using Z inference. Python output takes all of the math-work out of this for you.

Deviance

Deviance is a big deal. Let's set up some notation:

$\tilde{\beta}$: The MLE for model parameters in the ***fully saturated model*** (ie, the perfectly fit model.)

$\hat{\beta}$: The MLE for model parameters in ***your*** model.

$l(\hat{\beta}), l(\tilde{\beta})$: The likelihood functions of these models

Deviance

Then, the *deviance* is defined as follows:

$$D = 2\hat{\phi}\left[l(\tilde{\beta}) - l(\hat{\beta})\right] > 0$$

By definition, has highest likelihood

Must be lower, since you are using less information

So, the deviance sort of measures how much "information" you lose by fitting something besides the perfect model. That is, *lower deviance is better*. You can think of it as an MSE analog.

Recall $\hat{\phi}$ is from the Aitken EF form. Usually, you don't need to worry about it:

	Normal	Poisson	Bernoulli	Gamma
ϕ	σ^2	1	1	$1/\nu$

Deviance

However, the deviance doesn't actually have an interpretation on its own. And it's scale is different depending on your data. For example: what does a deviance of 243.8 mean?

So, we really only use it to compare to the deviances of other models.

Example: Model 2 is a reduced version of Model 1:

$$D_2 - D_1 \sim \chi^2_{p_1 - p_2}$$

An analysis of this can tell you if Model 2 is significantly different from Model 1.

Generalized Pearson's χ^2

In addition to the deviance, there is another measure of model fit, the *generalized Pearson's χ^2*:

$$\mathrm{X}^2 = \sum \frac{(y_i - \hat{y}_i)^2}{V(\hat{y}_i)} \sim \chi_p^2$$

Where $V(\mu)$ is the "variance function" from the Aitken EF form. Here it is for a few common GLMs:

	Normal	Poisson	Bernoulli	Gamma
$V(\mu)$	1	μ	$\mu(1-\mu)$	μ^2

GOTO: Code-a-long: Part I

https://github.com/TimothyKBook/ga-glm-lecture

The Exciting Stuff (GLMs for Classification!)

0/1 Binomial Logistic Regression

- **You might already be familiar with this.**
- **Your predictions are on a *probability scale*.**

0.32	0.84	0.54	0.12	0.37	0.66	0.03

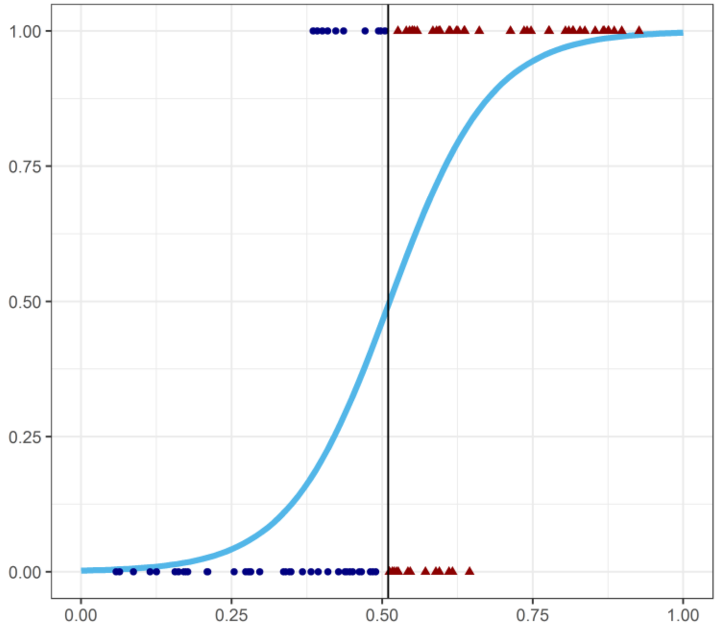
- **Can use a cutoff (usually ½), and threshold predictions to be 0/1.**

0	**1**	**1**	**0**	**0**	**1**	**0**

Example: Fake data from earlier

What is the benefit of using this method?

- It is usually better, or just as good as some of the “fancier” stuff like discriminant analysis and SVMs.
- Computationally cheap.
- Interpretable (can’t really explain an SVM, neural net, or RF to “normal people”).



Multiclass Classification

- **As I mentioned earlier, the *multinomial distribution* is legal tender for GLMs.**
- **Recall: The multinomial distribution works just like the binomial distribution, but there are multiple outcomes, rather than just 0/1. It's usually written:**

$$Y_i \sim \text{Mult}(n, \mathbf{p}_i)$$

Where $\mathbf{p}_i$ is a vector of probabilities for each class.

Multiclass Classification

So, for a GLM with K possible outcomes, we need to rethink our notation a bit.

$$Y_i \sim \text{ind Mult}(1, \mathbf{p}_i)$$

$$\log \frac{p_1}{p_K} = \beta_{01} + \beta_{11}x_1 + \cdots + \beta_{p1}x_p$$

$$\log \frac{p_2}{p_K} = \beta_{02} + \beta_{12}x_1 + \cdots + \beta_{p2}x_p$$

$$\ldots$$

$$\log \frac{p_{K-1}}{p_K} = \beta_{0K-1} + \beta_{1K-1}x_1 + \cdots + \beta_{pK-1}x_p$$

Notice we now have $(\boldsymbol{p}+\mathbf{1}) \times (\boldsymbol{K}-\mathbf{1})$ parameters to fit!

Multiclass Classification

For prediction, we solve a system of equations, which your computer can do for you. So, for the first time, *each observation $\mathbf{x}_i$ results in multiple predictions*: $(\hat{p}_1, \dots, \hat{p}_{K-1})$.

Each $\hat{p}_j$ represents the predicted probability that a given $\mathbf{x}_i$ falls into class j.

So, you can *classify* $\mathbf{x}_i$ into which ever has the maximum probability!

Multiclass Classification

Again, in the real world, this actually usually works better than the fancy stuff!

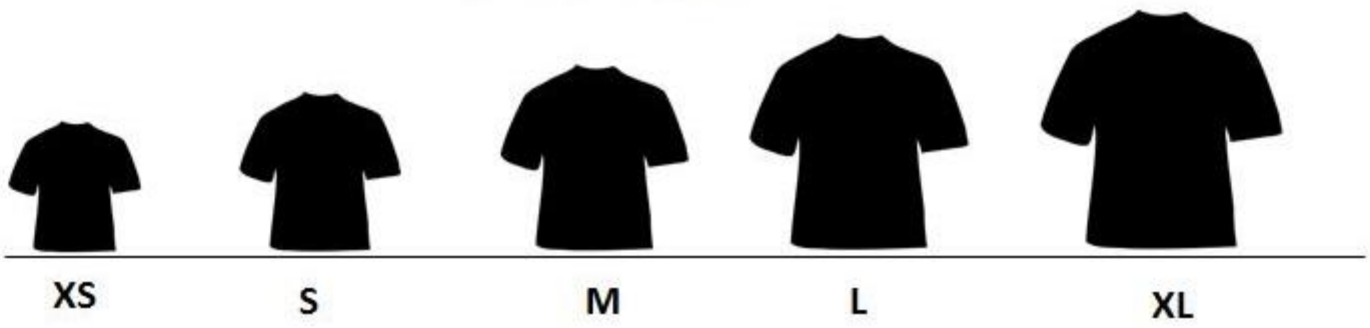
Want proof?

I used a multinomial logistic regression for my Kaggle submission. I spent about 20 minutes on this. And I even forgot to submit my best one!

I told you I was prolific and world famous.

Ordinal Regression

- **Suppose you have a multi-class response again, but this time your data are *ordinal*. That is, they aren't numbers, but they definitely have an order.**
  - Example: Small, Medium, Large, X-Large T-shirt sizes.



- **How can we tweak multinomial regression to handle this?**

Ordinal Regression

- **Let's define γ_j to be the *cumulative probability* of being in category j. That is,**

$$\gamma_j = \sum_{k=1}^{j} p_k = \mathrm{P}(\text{We are in category} \leq j)$$

The Cumulative Logit Model

Formally, we can define the *cumulative logit model* as:

$$Y_i \sim \text{ind Mult}(1, \mathbf{p})$$
$$\gamma_j = p_1 + \cdots + p_j$$

$$\text{logit}\, \gamma_1 = \beta_{01} + \beta_{11} x_i + \cdots + \beta_{p1} x_i$$
$$\text{logit}\, \gamma_2 = \beta_{02} + \beta_{12} x_i + \cdots + \beta_{p2} x_i$$
$$\ldots$$
$$\text{logit}\, \gamma_{K-1} = \beta_{0K-1} + \beta_{1K-1} x_i + \cdots + \beta_{pK-1} x_i$$

Cumulative Logit

And *prediction* can be done recursively:

$$p_1 = \gamma_1$$

$$p_j = \gamma_j - \gamma_{j-1}$$

$$p_K = 1$$

GOTO: Code-a-long Part II

https://github.com/TimothyKBook/ga-glm-lecture

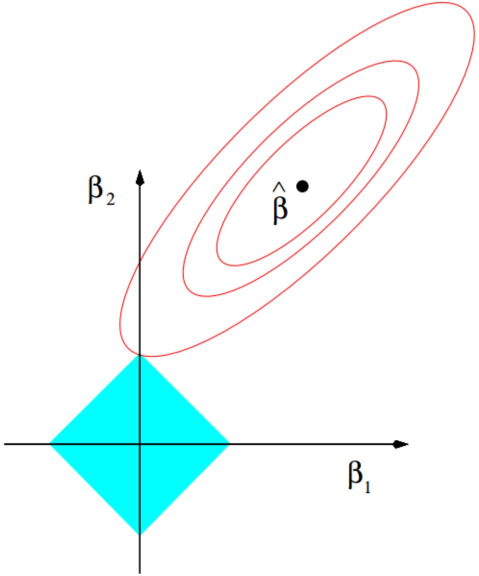
GLM Pot Pourri

Regularization

You might remember from last week the concept of *regularization* (or *penalized regression*).

Cliffsnotes: We want to optimize the likelihood, *but not that optimally*. For the *LASSO*, we minimize the following function:

$$\text{stuff} + \lambda ||\beta||_1$$



Regularization

Well – you guessed it. You can regularize GLMs.

If you attempted to fit a logistic regression using the `sklearn.LogisticRegression()` method, it actually tried to regularize by default, which is *annoying*.

Otherwise, you can use `sm.GLM().fit_regularized()`.

Generalized Generalized Linear Models

Yes you read that right. What if you think your response variables are actually correlated? You can fit a *Generalized Estimating Equation* (*GEE*).

GEEs can be fit in `statsmodels.gee()`.

WARNING: These take a *long* time to fit. Like, more than 10 minutes even for a small data set.

Overdispersion

- **Remember that all of these models also assumed *constant variance* among residuals. But…**
- **If** $X \sim \mathrm{Poi}(\lambda)$, **then** $\mathrm{E}[X] = \lambda$ **and** $\mathrm{Var}[X] = \lambda$
- **However, maximum likelihood methods often predict model variance to be *larger* than the response variable's mean.**
- **This effect can carry over into some GLMs. It's called *overdispersion*, and it's surprisingly common.**
- **An easy way to detect overdispersion is if the *residual degrees of freedom > deviance*.**

Overdispersion

- **How can we fix this? There are a few methods:**
  - Some people propose using a different link function. This isn't really a fix, though. But it works sometimes.
  - You can assume a more general variance structure. This can be done with ***quasi-likelihood*** methods. These are included in `statsmodels.GLM()`. (Similar to GEEs, these take a long time to fit).
  - You can use ***Bayesian methods***, if you really want…

Thank you guys for having me! Any Questions?