CSCI 3022-002 Intro to Data Science
Logistic Regression

Canvas:

see 3 years' past finals w/ solutions

Final:

Same structure

Same turn-in (Gradescope)

as midterm.

Announcements and Reminders

- ~~Homework due tonight!~~ see slide 1: Exam 2 ☹
- Kinda-sorta-optional notebook day ~~Wednesday~~ rest of semester. Practicum en route

ANOVA

An ANOVA was the linear models' way to capture *discrete* predictor variables. It does so by asking: do observations in one *group* of that predictor differ from the other groups? To compare A, B, and *neither*, this is a linear model of:

$$y_i = \beta_0 + \beta_1 \mathbb{1}_{x_i \in A} + \beta_2 \mathbb{1}_{x_i \in B} + \varepsilon$$

We call β_1 and β_2 **treatment effects.**

An ANOVA results in an F-statistic for whether or not the means of the groups differ. *If* this F-test meets our threshold for significance, we can follow it up to ask *which* groups differ from one another by a **Tukey** test, or Tukey HSD. We saw this in the notebook last Wednesday.

Where we at?

Our full MLR workflow:

Step	Idea	Plots	Fixes
0	Explore	pairs	just enjoy
1	Candidates	pairplot	remove redundant
2	Linearity	Residuals vs X/Y	transform?
3	Normality	Histograms and QQ	transform?
4	Homoskedasticity	Component-Residual	hard
5	Uncorrelation	Component-Residual	hard
6	Outliers	Influence, Cook's	remove?

log(x), e^x, ↗ polynomial in x.

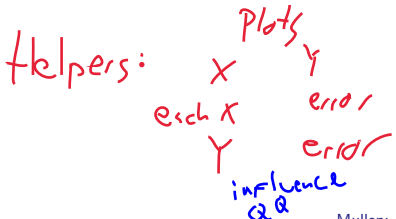
for each predictor x plot x = column, y = error/residual

We loop steps 2-6 until we're done, deciding whether or not our model is improving after each iteration by using things like SSE, adjusted R^2, and $F-$tests to compare between models.

Final steps: Re-run the model

Don't make too many changes at a time. Fix one or two things and re-run the model each time. What fixes one thing can and will lead to different points becoming outliers and often changes to every one of the diagnostics above.

Don't rush a regression problem. It's most a gradual process of not throwing out too much and making sure assumptions are met in the final model.



Extra plots

Python has some automated functions for most of these plots! In one of the in class notebooks, we make a FIT_AND_RES function that did residuals plots against a single predictor. statsmodels includes a few ways to automate some other commons plots. Given a model from MODEL=SM.OLS(Y,X).FIT()

1. pairs plots, via SEABORN.PAIRPLOT. **Visually** assess related X values, then check them numerically with SM.STATS.OUTLIERS_INFLUENCE.VARIANCE_INFLATION_FACTOR(X,I)
2. QQ plots to determine normality, via SM.QQPLOT(MODEL.RESID, STATS.T, DDOF). The the QQ plot isn't visually close to a straight line, it may mean that the errors are not normal. (it's plotting sample quantiles against theoretical t-critical value quantiles)
3. Component-residual plots. For a plot of X_i versus $resid$, you can use STATSMODELS.PLOT_REGRESS_EXOG(MODEL, I). Here we look for whether transformations or polynomials of *that* X should be used.
4. SM.GRAPHICS.INFLUENCE_PLOT(SLR, CRITERION="COOKS"). *Discuss* and **maybe** remove offenders

Final steps: Reporting your results

Your final writeup should include the following:

1. All terms in your final model, their estimates, and their confidence intervals. You may (probably should) also include a sentence interpreting each one and whether or not you find that result intuitive/reasonable. Keep in mind that β_j is the effect of the jth predictor given the inclusion of all of your other predictors.
2. The F-test p-value and whether or not you reject the associated hypothesis. What does this mean?
3. Discuss whether your final model includes any insignificant predictors (e.g. the t-tests for those predictors would not have you reject H_0.) Why are they in your model?
4. How useful is your model? Discuss R^2, and any questions about prediction.

Final steps: Choosing between Models

Let's say you've "tuned" 2 or 3 different models with different subsets of predictors included. How do we choose between them?

1. Best adjusted R^2
2. Best AIC or BIC.
3. Lowest Standard Error (least SSE)?
4. Only significant terms included?

Level 2: Automating the Choice between Models

If you want to do a quick search on the optimization between models (without stopping between each to tune, there are 3 common schema)

1. Test all subsets (how many computations is this?)
2. Forward inclusion (add the best "missing" term each step until there's no longer anything useful to add)
3. Backward inclusion

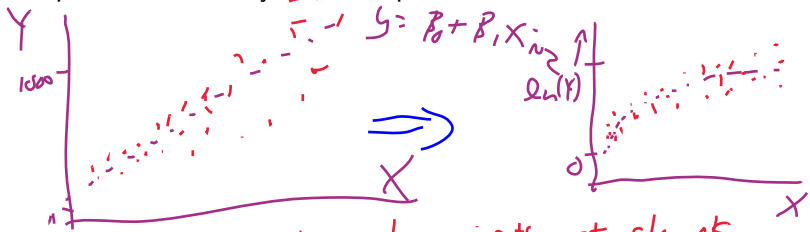
On Transformations

$Y_i = \beta_0 + \beta_1 X_i + \varepsilon$ (error)

what if $X_i \to \infty$?

At this point, our two biggest tools to fix problems with our models are either adding more columns (polynomial regression terms) or transforming columns.

We also have the option to transform Y. One main reason to transform Y over an X column is that the ε errors are Y errors: if Y for some reason has bounds: like a count or a percentage score or something else, the assumption of normality becomes problematic. So does a line!



Suppose.

if error grows with y: replace y with a function where larger inputs get shrunk faster:

Binary Outcomes

To date, we've dealt with continuous Y. What if Y were a binary outcome, like our Titanic data form long ago?

	age	outcome
0	25	survived
1	30	survived
2	35	survived
3	40	survived
4	45	died
5	50	died
6	55	died
7	60	died

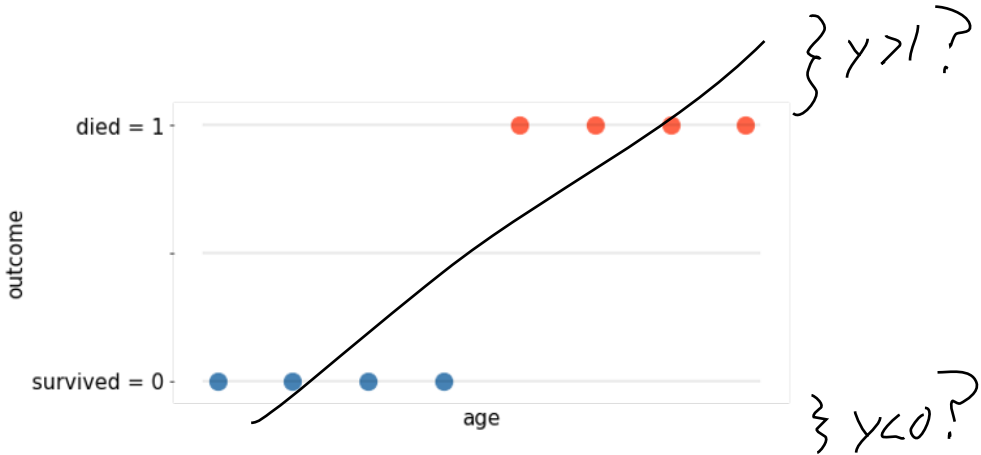
we rewrite as:

	age	outcome
0	25	0
1	30	0
2	35	0
3	40	0
4	45	1
5	50	1
6	55	1
7	60	1

What happens if we do a linear regression?

Binary Outcomes

Goal: predict whether a passenger survives or not as a linear function of their age.

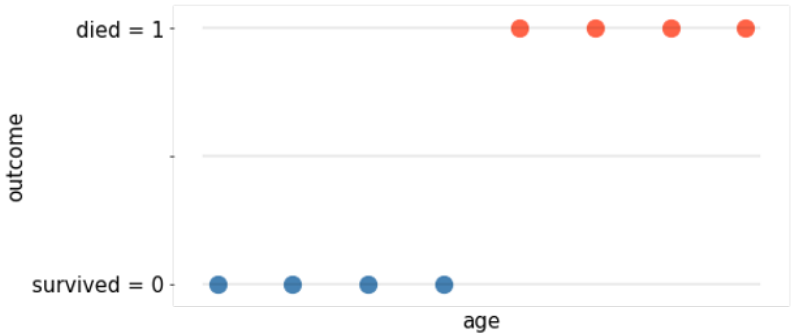


Binary Outcomes

Goal: predict whether a passenger survives or not as a linear function of their age.

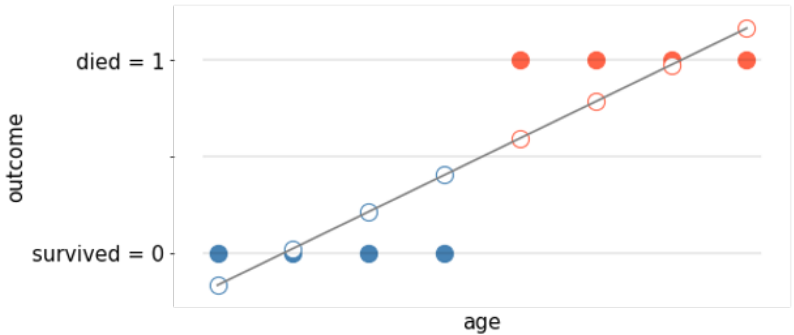
$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Single input of X :=age, output desired is a prediction of {0,1}={Survived, Died}



Binary Outcomes... as a line?

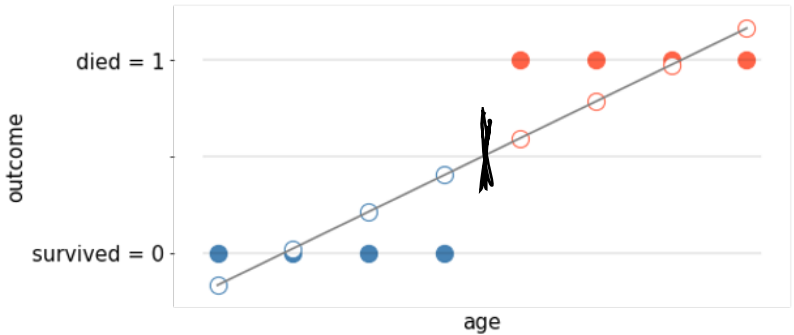
Goal: predict whether a passenger survives or not as a linear function of their age:



Binary Outcomes... as a line?

Goal: predict whether a passenger survives or not as a linear function of their age:
Did it work? We could use this line to do some kind of classification, e.g. use a piecewise function like

$$y = \begin{cases} 0 & \text{if } x < \text{threshold} \\ 1 & \text{if } x \geq \text{threshold} \end{cases}$$

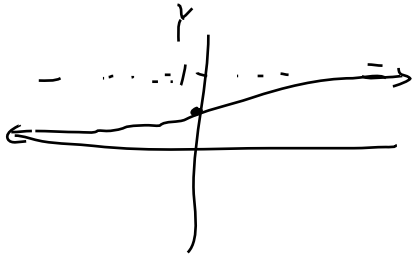


Binary Outcomes and error

Directly using Y here is a bit awkward though.

1. What do our errors represent in general?
2. How do we interpret linear fits less than zero and greater than 1?
3. We really want a measure here that behaves more like *probability.*

Binary Outcomes and error



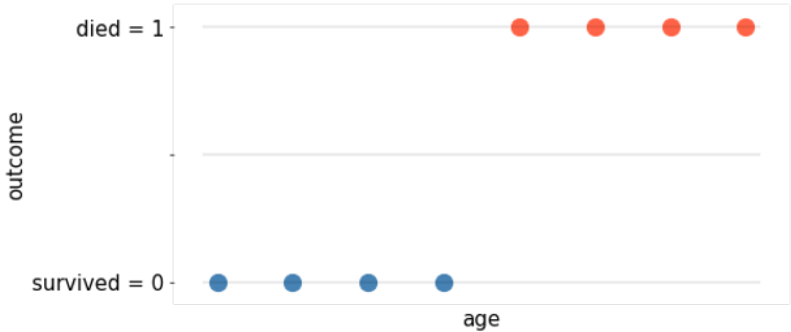
Directly using Y here is a bit awkward though.

1. What do our errors represent in general?
2. How do we interpret linear fits less than zero and greater than 1?
3. We really want a measure here that behaves more like *probability.*
4. Let's reach into the math toolbox... we need a function that an input (*domain*) of (0,1) and will give us a range of $\mathbb{R}$... or vice versa, and we can invert it.

if X Y(prob)
$-\infty$ 0
0 .5
∞ 1

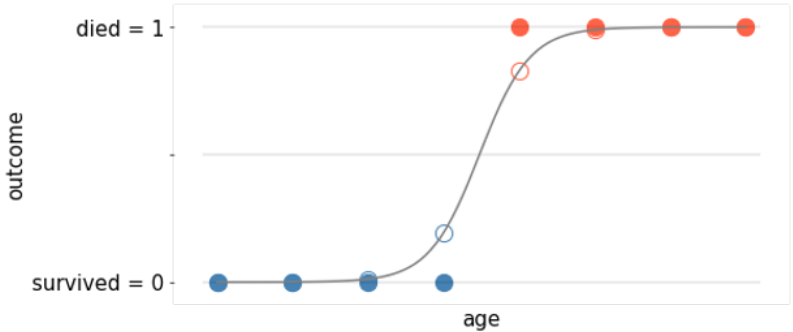
Binary Outcomes: a better fit

Goal: predict whether a passenger survives or not as a linear function of their age:
Let's sketch what we might want a probability-based function to look like:



The Sigmoid

Goal: predict whether a passenger survives or not as a linear function of their age:
How's this thing look?

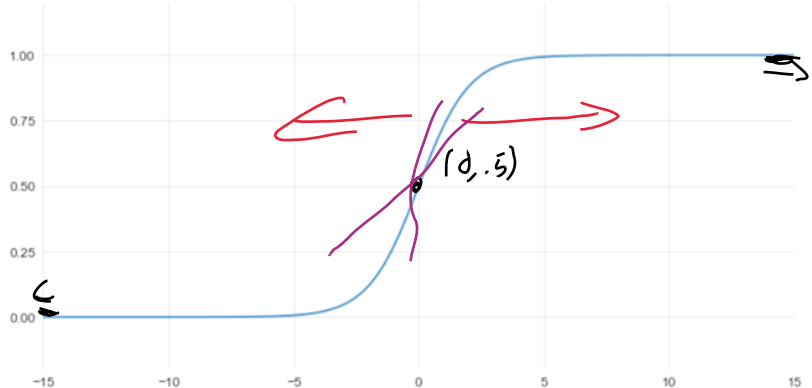


Sigmoid!

The Sigmoid

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

has the properties we want.



$e^{-(z-a)}$ (right by a)

e^{-bz} stretches/contracts factor of b.

How would we move it left-to-right? How would we make it steeper or shallower?

The Sigmoid

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

center · steep

1. We can use something like $\text{sigm}(\beta_0 + \beta_1 x)$: this gives us dials to both control where the sigmoid moves from 0 to 1 and how steeply it does so.
2. Because the output of the sigmoid is (0,1), we can treat it's outputs like probabilities.
3. It's really smooth. This means we're probably not naturally over-fitting!

The Sigmoid

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

1. We can use something like $\text{sigm}(\beta_0 + \beta_1 x)$: this gives us dials to both control where the sigmoid moves from 0 to 1 and how steeply it does so.
2. Because the output of the sigmoid is (0,1), we can treat it's outputs like probabilities.
3. It's really smooth. This means we're probably not naturally over-fitting!

Model:

$$P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$$

Finding our Sigmoid

Model:

$$P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$$

Plan:

learn the weights of $\beta_0 + \beta_1 X$ from the data by estimating them:
I.E. find suitable $\hat{\beta}_0$; $\hat{\beta}_1$

Classify: Gain the ability to describe point x according to

$$\hat{y} = \begin{cases} 1 & \text{if sigm}(\hat{\beta}_0 + \hat{\beta}_1 X) \geq .5 \\ 0 & \text{if sigm}(\hat{\beta}_0 + \hat{\beta}_1 X) < .5 \end{cases}$$

Finding our Sigmoid

$$Y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

we know best β_s for,

$y = \beta_0 + \beta_1 X_1$ (minimize squared error).

This sounds a lot like a regression problem! And it is! But we've come at it a little backwards. It turns out we could have arrived at the sigmoid by transforming Y and X in our initial problem.

More generally, a crucial interpretation of the sigmoid relies on treating Y as probability. Let's talk about *odds*.

Could instead set

$g(Y) = \beta_0 + \beta_1 X$

Odds

In statistics, the *odds* of an event occuring is defined as the ratio of the probability of an event occuring divided by the probability of it not occuring. It is often then flipped to be guaranteed larger than 1, at least for colloquial use. So

$$\text{Odds} = \frac{p}{1-p} \quad \text{or} \quad \frac{1-p}{p}$$

Example: If $p = .75$, the odds are $\frac{.75}{.25} = 3$; $\frac{3}{1}$ and we might say: 3:1.

Example: If $p = .1$, the odds are $\frac{.1}{.9}$; $\frac{1}{9}$ and we might say: 9:1 against

Odds

In statistics, the *odds* of an event occuring is defined as the ratio of the probability of an event occuring divided by the probability of it not occuring. It is often then flipped to be guaranteed larger than 1, at least for colloquial use. So

$$\text{Odds} =$$

Example: If $p = .75$, the odds are $\frac{3/4}{1/4} = 3$;
and we might say: 3 to 1 in favor.

Example: If $p = .1$, the odds are $\frac{.1}{.9} = \frac{1}{9}$;
and we might say: 9 to 1 against.

Odds are this is useful

In logistic regression, we're using a model of the probability p, or

$$p = P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$$

What does this mean in terms of odds?

$$\text{Odds} = \frac{p}{1-p}$$

Odds are this is useful

In logistic regression, we're using a model of the probability p, or

$$p = P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$$

What does this mean in terms of odds? (using z as our line, for shorthand)

$$\begin{aligned}\text{Odds} &= \frac{p}{1-p} \\ &= \frac{\frac{1}{1+e^{-z}}}{1 - \frac{1}{1+e^{-z}}}\end{aligned}$$

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

$= 1/e^{-z} = e^{z}$

Odds are this is useful

In logistic regression, we're using a model of the probability p, or

$$p = P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$$

What does this mean in terms of odds?

$$\begin{aligned}
\text{Odds} &= \frac{p}{1-p} \\
&= \frac{\frac{1}{1+e^{-z}}}{1 - \frac{1}{1+e^{-z}}} \\
&= \frac{1}{1 + e^{-z} - 1} = e^z = e^{\beta_0 + \beta_1 X}
\end{aligned}$$

Odds are this is useful

Taking the log of both sides gives

$$\ln \text{Odds} = \beta_0 + \beta_1 X$$

which is a simple linear regression problem!

Only this time, a one-unit increase in X represents an increase of β_1 in the *Log-Odds* of y.

If y had begun as a probability between 0 and 1 instead of exactly 0 or 1, we could have taken $y_{new} = \ln \frac{y}{1-y}$ as a transformation of y that would move it from (0,1) to (∞, ∞). This is the same intepretation as logistic regression, but we can't actually apply that transformation since the actual data values are 0 and 1!

Big Logistic Regression

The simple logistic regression model is then $P(Y = 1|X) = \text{sigm}(\beta_0 + \beta_1 X)$.

In reality, we will have many features or predictors. For example:

Predict the probability of precipitation or a storm

Given temperature, pressure, humidity, whether it rained yesterday (or a week ago!), etc.

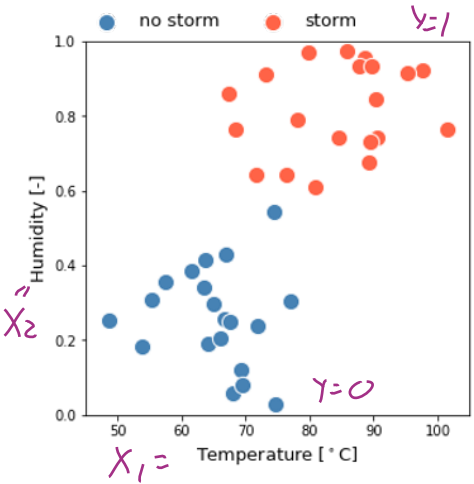
Big Logistic Regression

Predict Was there a storm (if storm then, $Y = 1$)?

Features X_1 :=Temp, X_2 :=humidity

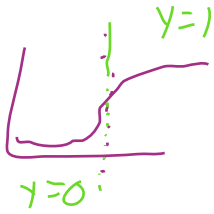
Model:

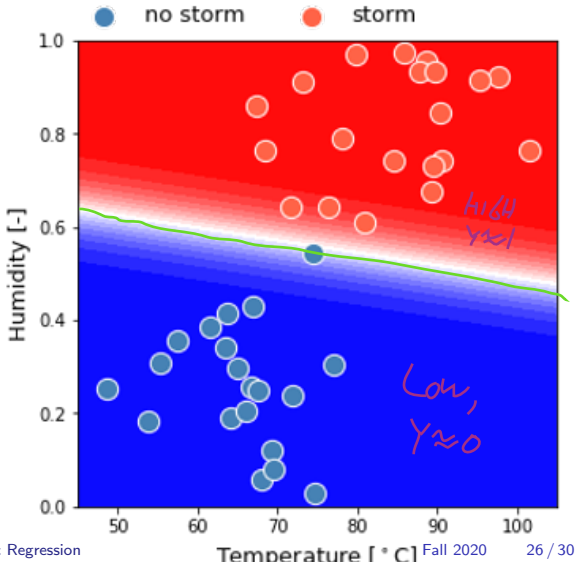
$$P(\text{Storm}) = \text{sigm}(\beta_0 + \beta_1 X_1 + \beta_2 X_2)$$



Prediction Surface

The result might be a *surface* of log-odds values which we could transform back to probabilities.

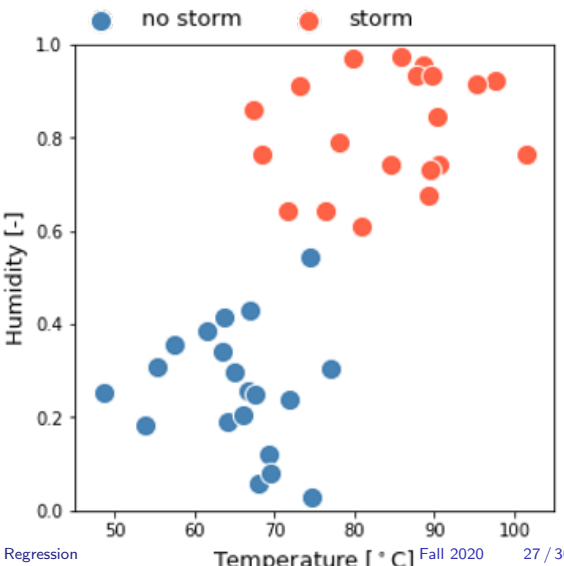




Prediction Boundary

Of course, we also might only be using the line that defines the boundary: that's the criteria we use to decide whether our best guess $\hat{y} = 1$ or not.

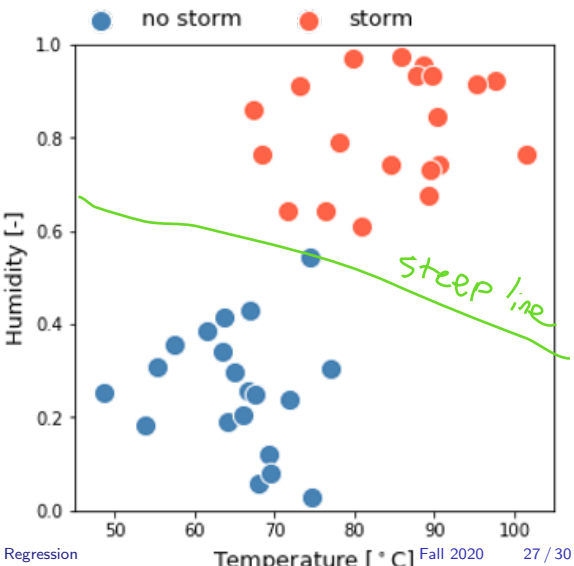
The sigmoid function happens to have a nice property that $f'(z) = f(z)(1 - f(z))$, which can make for very useful calculus!



Prediction Boundary

Of course, we also might only be using the line that defines the boundary: that's the criteria we use to decide whether our best guess $\hat{y} = 1$ or not.

The sigmoid function happens to have a nice property that $f'(z) = f(z)(1 - f(z))$, which can make for very useful calculus! **Steepest** at .5... sounds useful.



Prediction Boundary: See ya later

The goal of finding a prediction boundary: like a line that might satisfy some appealing properties is a major algorithmic concern in CSCI4622: Machine Learning.

In that class, we rewrite the problem as "how do we best draw a line (or (hyper-)plane! to separate the 0's and the 1's." Which side of the line a new point falls on becomes our classifier. This is called a support-vector machine (SVM).

Data Science, extended

Many data science problems have multiple ways to answer the same problem. For just this classification problem, we could have arrived at a classifier by drawing an optimal sigmoid (nonlinear least-squares), doing an SVM, or doing transformations of probability data y into log-odds and doing regression.

In general, most data science problems boil down to two or three things:

1. What's the domain problem? Why do we care?
2. What's the thing you need to estimate? Slopes? Outcomes? Means?
3. How do you calculate that estimator? Calculus? Brute Force?

As you move on, you'll have to choose between a variety of similar and overlapping techniques according to what **assumptions** lie in #2 and what **computational costs** lie in #3.

So, where next?

Enjoy this course? Some sequels: (SQLs?)

1. CSCI3202: Intro to AI (R or Python, depending)
   - The bridge between probability and "machine learning." Leads to lots of classification schemes (trees, SVMs, NNs) and "deep learning." Taught by Chris Heckman.
2. CSCI4022: Advanced Data Science (Python)
   - Techniques and algorithms in unsupervised in machine learning, similarity measures, dimension reduction. Lots of maximum likelihood! Taught by me!
3. STAT4010: Stat. Methods and Applications II. (R)
   - Linear modeling, part deux. How to start dealing with autocorrelation, more complicated ANOVAs, and doing Bayesian estimation of linear models (*wow!*). Taught by Brian Zaharatos (brian.zaharatos@colorado.edu)