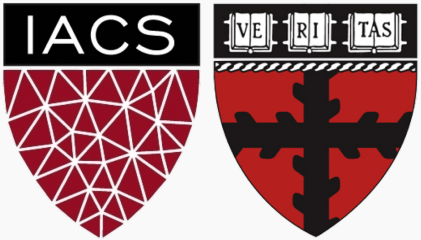
Lecture 15: *k*-NN Classification & Logistic Regression

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner



Announcements

0. HW3 Due
1. **HW4** released later today
   - Due in 2 weeks (Oct 21)
   - Individual

2. **Project** topic assignments
   - Released tomorrow (every group gets assigned a TF)
   - Every full group got their first choice
   - Those with custom topics will take a day or two more processing time.
3. No **Lecture** on Monday! No **Sections** Friday or Monday. No OHs on Monday.
4. Next **Advanced Section**: next week (Oct 14)

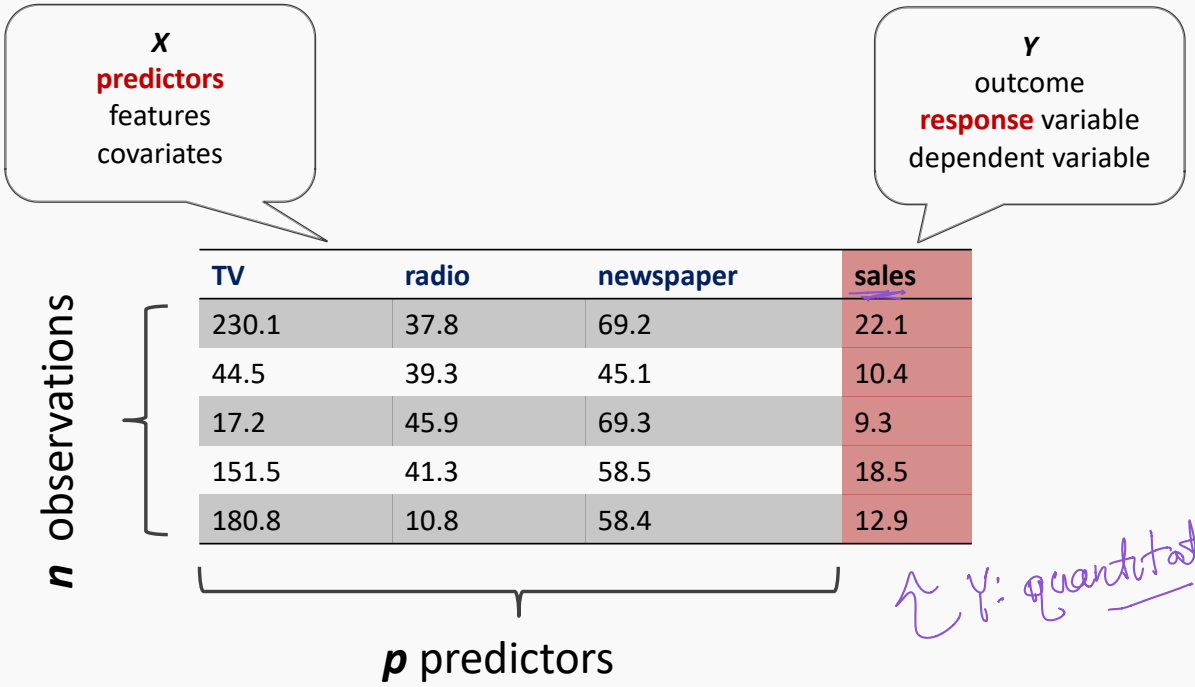
↳ G.L.M.

Lecture Outline

- Introduction to Classification
- *k*-NN for Classification
- Why not Linear Regression?
- Binary Response & Logistic Regression
  - Estimating the Simple Logistic Model
  - Classification using the Logistic Model
  - Multiple Logistic Regression

Classification

Advertising Data (from earlier lectures)



TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

Y: quantitative

Heart Data

categorical w/ K "classes"

response variable ***Y*** is Yes/No

Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

Heart Data

These data contain a binary outcome HD for 303 patients who presented with chest pain. An outcome value of:

- ***Yes*** indicates the presence of heart disease based on an angiographic test,
- ***No*** means no heart disease.

There are 13 predictors including:

- Age
- Sex (0 for women, 1 for men)
- Chol (a cholesterol measurement),
- MaxHR
- RestBP

and other heart and lung function measurements.

Classification

Up to this point, the methods we have seen have centered around modeling and the prediction of a **quantitative** response variable (ex, number of taxi pickups, number of bike rentals, etc). Linear **regression** (and Ridge, LASSO, etc) perform well under these situations

When the response variable is **categorical**, then the problem is no longer called a regression problem but is instead labeled as a **classification problem**.

The goal is to attempt to classify each observation into a category (aka, class or cluster) defined by Y, based on a set of predictor variables X.

Typical Classification Examples

The motivating examples for the lecture(s), homework, and section are based [mostly] on medical data sets. Classification problems are common in this domain:

- Trying to determine where to set the *cut-off* for some diagnostic test (pregnancy tests, prostate or breast cancer screening tests, etc...)
- Trying to determine if cancer has gone into remission based on treatment and various other indicators
- Trying to classify patients into types or classes of disease based on various genomic markers

k-NN for Classification

k-Nearest Neighbors

We've already seen the *k*-NN method for predicting a quantitative response (it was the very first method we introduced). How was *k*-NN implemented in the Regression setting (quantitative response)?

The approach was simple: to predict an observation's response, use the **other** available observations that are most similar to it.

For a specified value of *k*, each observation's outcome is predicted to be the **average** of the *k*-closest observations as measured by some distance of the predictor(s).

With one predictor, the method was easily implemented.

Review: Choice of k

small k: complex model

How well the predictions perform is related to the choice of *k*.

large k: "simple" model

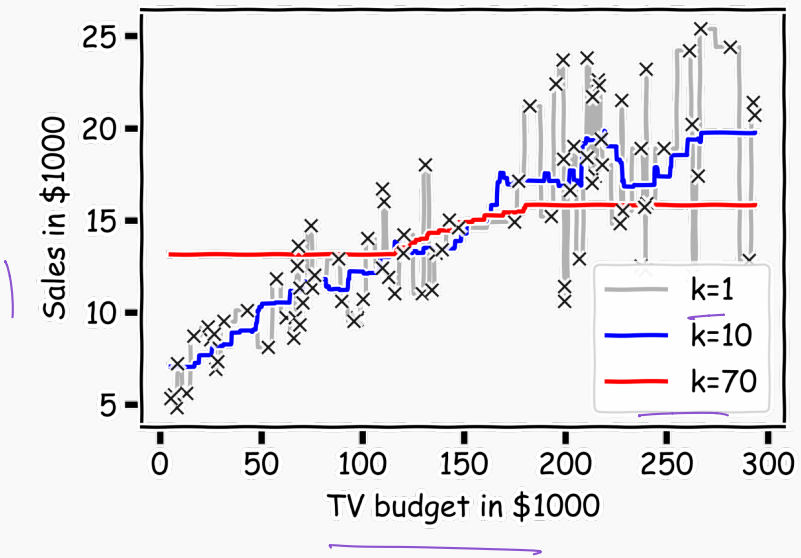
What will the predictions look like if *k* is very small? What if it is very large?

More specifically, what will the predictions be for new observations if $k = n$?

$\bar{Y}$

A picture is worth a thousand words...

Choice of *k* matters



k-NN for Classification

How can we modify the *k*-NN approach for classification?

The approach here is the same as for *k*-NN regression: use the other available observations that are most similar to the observation we are trying to predict (classify into a group) based on the predictors at hand.

How do we classify which category a specific observation should be in based on its nearest neighbors?

"majority" or "plurality" wins

k-NN for Classification: formal definition

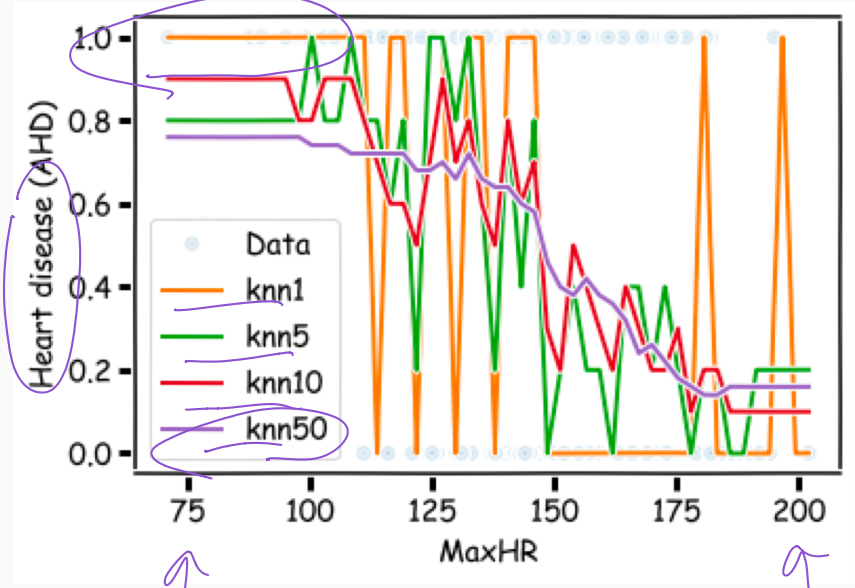
The k-NN classifier first identifies the k points in the training data that are closest to x_0, represented by $\mathcal{N}_0$. It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$P(Y = j | X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

proportion in each class in the neighborhood

Then, the k-NN classifier predicts this new observation, x_0, to be in the class with largest estimated probability.

Estimated Probabilities in *k*-NN Classification



predicted probabilities of $y=1$

$\hat{p}(y=1)$

k-NN for Classification (cont.)

There are some issues that may arise:

- How can we handle a tie?

flip a coin

- What could be a major problem with always classifying to the most common group amongst the neighbors?

unbalanced classes?

0.99 in group 0
0.01 in group 1

- How can we handle this?

↳ over sample or overweight the less common classes.

k-NN Classification in Python

Performing kNN classification in python is done via **`KNeighborsClassifier`** in **`sklearn.neighbors`**.

An example:

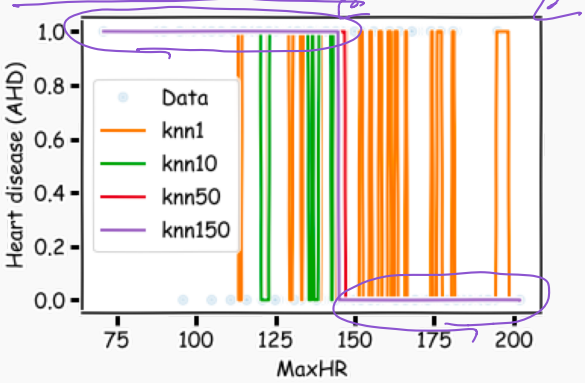
```
data_x = df_heart[['MaxHR']]
data_y = df_heart['AHD']

knn1 = KNeighborsClassifier(n_neighbors=1).fit(data_x, data_y)
knn10 = KNeighborsClassifier(n_neighbors=10).fit(data_x, data_y)
knn50 = KNeighborsClassifier(n_neighbors=50).fit(data_x, data_y)
knn150 = KNeighborsClassifier(n_neighbors=150).fit(data_x, data_y)

print(knn1.score(data_x,data_y))
print(knn10.score(data_x,data_y))
print(knn50.score(data_x,data_y))
print(knn150.score(data_x,data_y))
```

Classification Accuracy

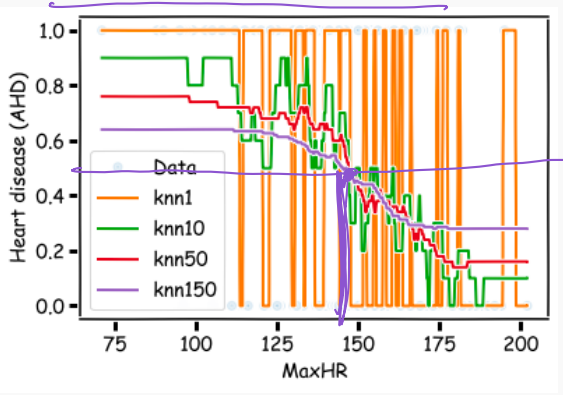
```
0.693069306930693
0.7161716171617162
0.7062706270627063
0.6996699669966997
```

Pure Classifications in *k*-NN Classification Models

Pure Classifications:



Estimated Probabilities:



What will these plots look like when there are 2 predictors? Or 3+ predictors?

This is our first glimpse at a **classification boundary** (more to come).

k-NN with Multiple Predictors

How could we extend *k*-NN (both regression and classification) when there are multiple predictors?

We would need to define a measure of distance for observations in order to which are the most similar to the observation we are trying to predict.

Euclidean distance is a good option. To measure the distance of a new observation, $\boldsymbol{x}_0$ from each observation in the data set, $\boldsymbol{x}_i$:

$$D^2(\boldsymbol{x}_i, \boldsymbol{x}_0) = \sum_{j=1}^{P} \left(x_{i,j} - x_{0,j}\right)^2$$

scale of predictors is important.

k-NN with Multiple Predictors (cont.)

But what must we be careful about when measuring distance?

1. Differences in variability in our predictors!
2. Having a mixture of quantitative and categorical predictors.

So what should be good practice? To determine closest neighbors when $p > 1$, you should first standardize the numeric predictors! And you can even standardize the binaries if you want to include them.

How else could we determine closeness in this multi-dimensional setting?

Parametric Modeling: Why not Linear Regression?

Simple Classification Example

Given a dataset:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$$

where the y are categorical (sometimes referred to as ***qualitative***), we would like to be able to predict which category y takes on given x.

A categorical variable y could be encoded to be quantitative. For example, if y represents concentration of Harvard undergrads, then y could take on the values:

$$y = \begin{cases} 1 & if \text{ Computer Science (CS)} \\ 2 & if \text{ Statistics} \\ 3 & \text{otherwise} \end{cases}$$

$K = 3$ classes

Linear regression **does not work well**, or is not appropriate at all, in this setting.

Simple Classification Example (cont.)

A linear regression could be used to predict *y* from ***x***. What would be wrong with such a model?

The model would imply a specific ordering of the outcome, and would treat a one-unit change in *y* equivalent. The jump from *y* = 1 to y = 2 (CS to Statistics) should not be interpreted as the same as a jump from *y* = 2 to *y* = 3 (Statistics to everyone else).

Similarly, the response variable could be reordered such that *y* = 1 represents Statistics and *y* = 2 represents CS, and then the model estimates and predictions would be fundamentally different.

If the categorical response variable was ***ordinal*** (had a natural ordering, like class year: Freshman, Sophomore, etc.), then a linear regression model would make some sense but is still not ideal.

Even Simpler Classification Problem: Binary Response

The simplest form of classification is when the response variable y has only two categories, and then an ordering of the categories is natural. For our example, a patient in the ICU could be categorized as having [atherosclerotic] heart disease (AHD) or not (note, the $y = 0$ category is a "catch-all" so it would involve those patients with lots of other diseases or diagnoses):

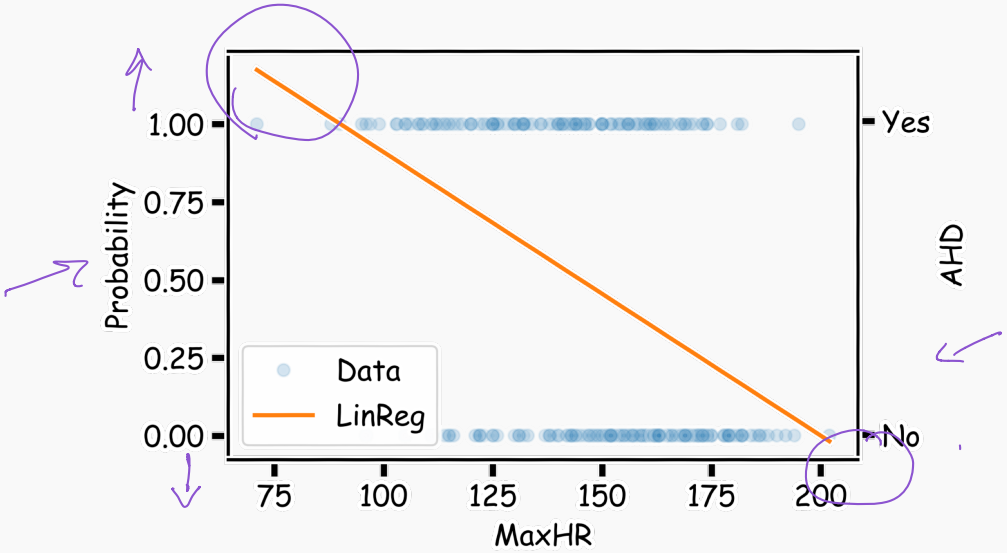
$$y = \begin{cases} 1 \text{ if patient has heart disease} \\ 0 \text{ otherwise.} \end{cases}$$

Linear regression could be used to predict y directly from a set of covariates (like sex, age, resting HR, etc.), and if $\hat{y} \geq 0.5$, we could predict the patient to have AHD and predict not to have heart disease if $\hat{y} < 0.5$.

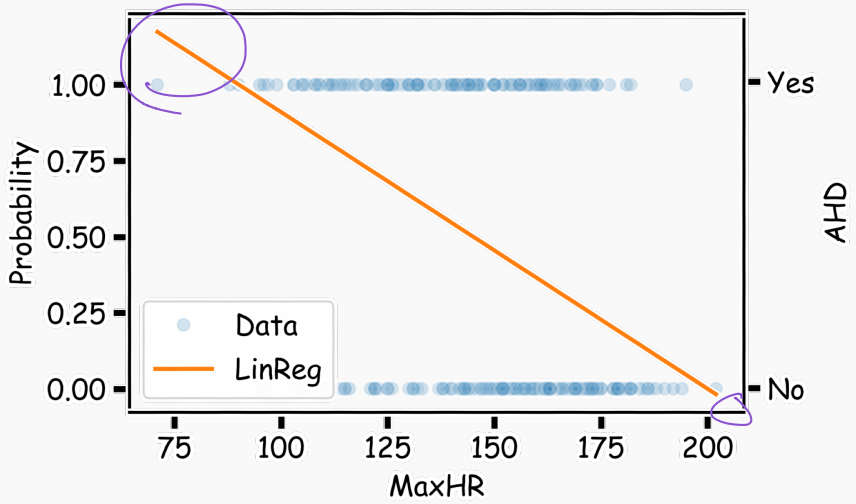
Even Simpler Classification Problem: Binary Response (cont)

What could go wrong with this linear regression model?

.



Even Simpler Classification Problem: Binary Response (cont)

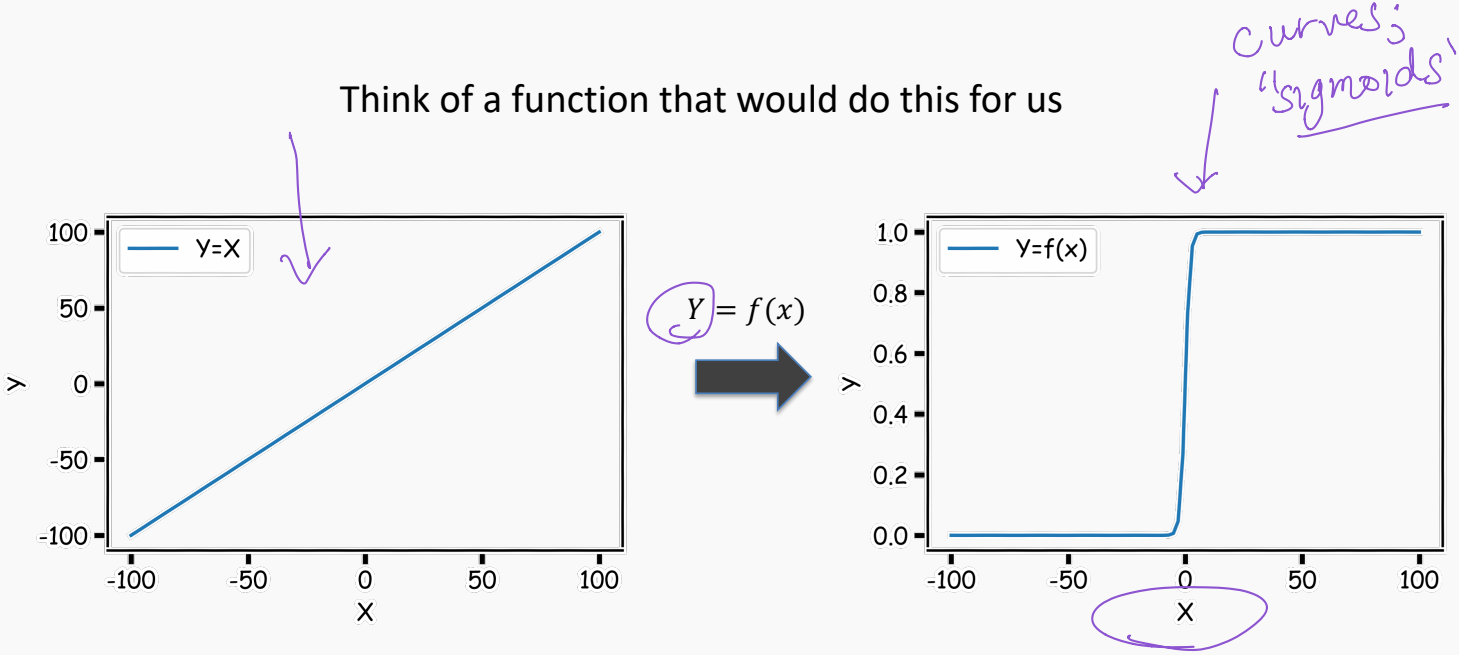


The main issue is you could get non-sensical values for y. Since this is modeling $P(y = 1)$, values for $\hat{y}$ below 0 and above 1 would be at odds with the natural measure for y. Linear regression can lead to this issue.

Binary Response & Logistic Regression

Pavlos Game #45

Think of a function that would do this for us



Logistic Regression

Logistic Regression addresses the problem of estimating a probability, $P(y = 1)$, to be outside the range of $[0,1]$. The logistic regression model uses a function, called the ***logistic*** function, to model $P(y = 1)$:

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

linear portion

$-\infty \Rightarrow P(Y=1) \to 0$

$\infty \Rightarrow P(Y=1) = 1$

Logistic Regression

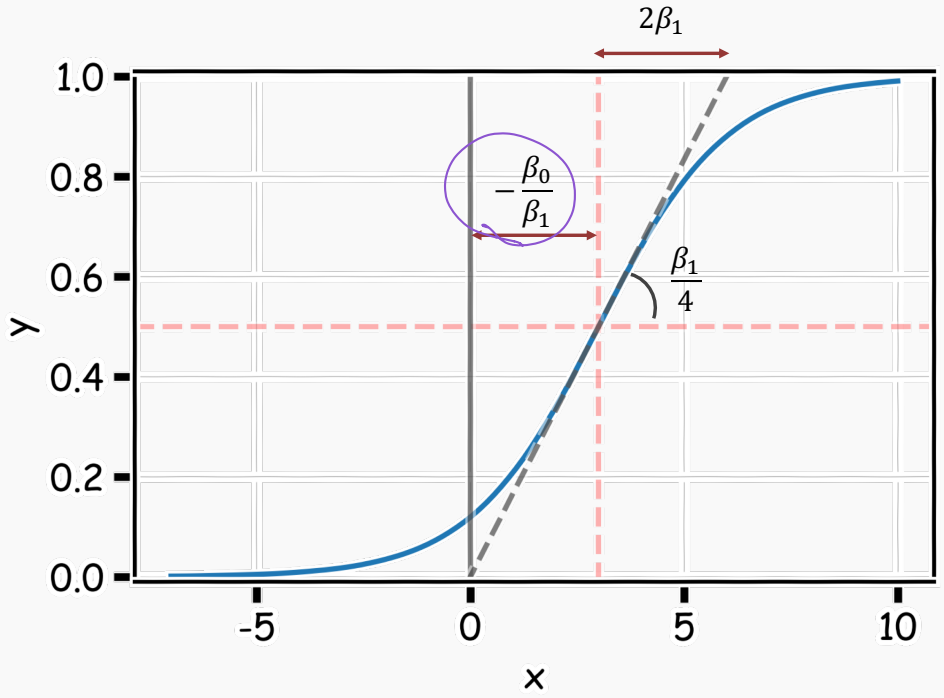
As a result the model will predict $P(y = 1)$ with an S-shaped curve, which is the general shape of the logistic function.

β_0 shifts the curve right or left by $c = -\frac{\beta_0}{\beta_1}$.

β_1 controls how steep the S-shaped curve is. Distance from ½ to almost 1 or ½ to almost 0 is $\frac{2}{\beta_1}$

Note: if β_1 is positive, then the predicted $P(y = 1)$ goes from zero for small values of X to one for large values of X and if β_1 is negative, then the $P(y = 1)$ has opposite association.

Logistic Regression

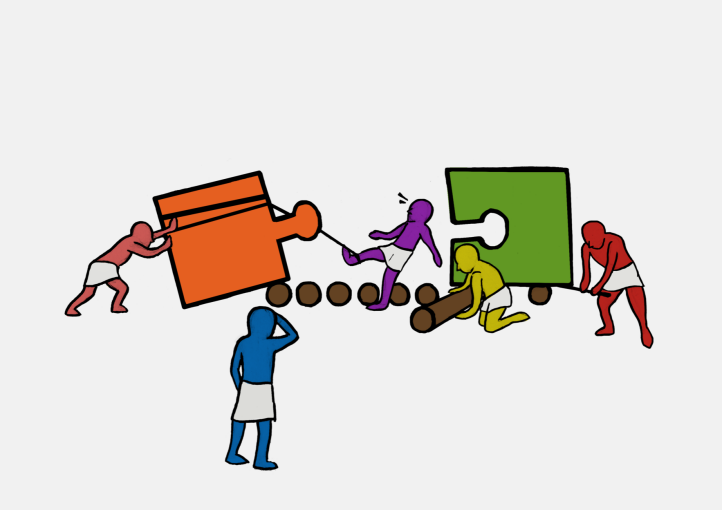


Logistic Regression: interpretation

With a little bit of algebraic work, the logistic model can be rewritten as:

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X.$$

The value inside the natural log function $\frac{P(Y=1)}{1-P(Y=1)}$, is called the ***odds***, thus logistic regression is said to model the ***log-odds*** with a linear function of the predictors or features, X. This gives us the natural interpretation of the estimates similar to linear regression: a one unit change in X is associated with a β_1 change in the log-odds of *success* $(Y = 1)$; or better yet, **a one unit change in X is associated with an e^{β_1} multiplicate change in the odds of *success* $(Y = 1)$** .



Exercise Time!

Ex. 1: Guesstimating the logistic function (10 min)

Estimating the Simple Logistic Model

Estimation in Logistic Regression

Unlike in linear regression where there exists a closed-form solution to finding the estimates, $\hat{\beta}_j$'s, for the true parameters, logistic regression estimates cannot be calculated through simple matrix multiplication.

Questions:

- In linear regression what loss function was used to determine the parameter estimates?
- What was the probabilistic perspective on linear regression?
- Logistic Regression also has a likelihood based approach to estimating parameter coefficients.

Estimation in Logistic Regression

Probability Mass Function (PMF):

$$P(Y = 1) = p$$
$$P(Y = 0) = 1 - p$$

$$P(Y = y) = p^y (1 - p)^{(1-y)}$$

where:

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

and therefore p depends on X.

Thus not every p_i is the same for each individual measurement.

Likelihood

The likelihood of a single observation for *p* given *x and y* is:

$$L(p_i|Y_i) = P(Y_i = y_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

Given the observations are independent, what is the likelihood function for *p*?

joint likelihood of all n observations

product

$$L(p|Y) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i}(1-p_i)^{1-y_i}$$

"ln"

$$l(p|Y) = -\log L(p|Y) = -\sum_i y_i \log p_i + (1-y_i)\log(1-p_i)$$

loss function

minimize negative log-likelihood

Loss Function

$$l(p|Y) = -\sum_i \left[y_i \log \frac{1}{1+e^{-(\beta_0+\beta_1 X_i)}} + (1-y_i) \log\left(1 - \frac{1}{1+e^{-(\beta_0+\beta_1 X_i)}}\right)\right]_i$$

How do we minimize this?

Differentiate, equate to zero and solve for it!

But jeeze does this look messy?! It will not necessarily have a closed form solution.

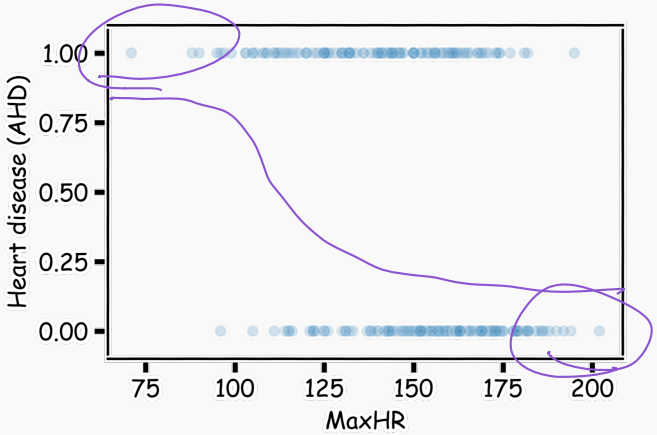
So how do we determine the parameter estimates? Through an iterative approach (we will talk about this *at length* in future lectures).

gradient descent

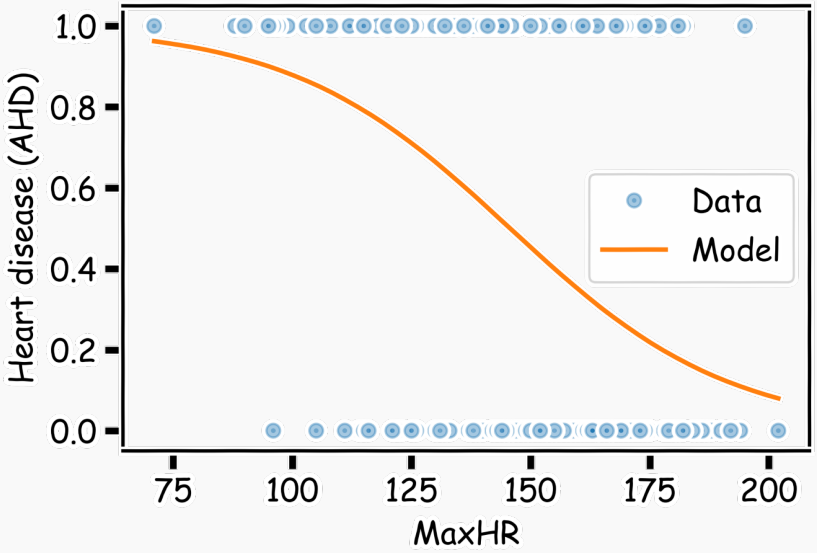
Heart Data: logistic estimation

We'd like to predict whether or not a person has a heart disease. And we'd like to make this prediction, for now, just based on the MaxHR.

How should we visualize these data?



Heart Data: logistic estimation



Heart Data: logistic estimation

There are various ways to fit a logistic model to this data set in Python. The most straightforward in `sklearn` is via `linear_model.LogisticRegression`.

```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C=100000, fit_intercept=True)
logreg.fit(data_x.values.reshape(-1,1), data_y);


print('Estimated beta1: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1:
 [[-0.04326016]]
Estimated beta0:
 [ 6.30193148]
```

function has regularization by default

$C = 1/\lambda$

penalty = "none"

↑ unregularized

$$\ln\left(\frac{P(Y=1)}{P(Y=0)}\right) = 6.302 - 0.043(X)$$

X: maxHR

Heart Data: logistic estimation

Answer some questions:

- Write down the logistic regression model.
- Interpret $\hat{\beta}_1$.
- Estimate the probability of heart decease for someone (like Pavlos) with MaxHR≈200?
- If we were to use this model purely for classification, how would we do so? See any issues?

odds of AHD

$e^{\hat{\beta}_1}$ ⇒ estimated multiplicative change in odds for a 1-unit change maxHR

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

$= 0.5$

Categorical Predictors

Just like in linear regression, when the predictor, X, is binary, the interpretation of the model simplifies (and there is a quick closed form solution here).

In this case, what are the interpretations of $\hat{\beta}_0$ and $\hat{\beta}_1$?

For the heart data, let X be the indicator that the individual is a female (X = 0) or male (X = 1). What is the interpretation of the coefficient estimates in this case?

$$\ln(\text{odds}) = \beta_0 + \beta_1 X$$

The observed percentage of HD for women is 26% while it is 55% for men.

Calculate the estimates for $\hat{\beta}_0$ and $\hat{\beta}_1$ if the indicator for HD was predicted from the gender indicator (work on next page).

$$\beta_0 = \ln\left(\frac{0.26}{0.74}\right) \qquad \beta_1 = \ln\left(\frac{0.55}{0.45}\right) - \beta_0$$

Solving for the estimates mathematically:

$$P(Y = 1|X = 0) = 0.26, \qquad P(Y = 1|X = 1) = 0.55$$

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 X$$

$$\ln\left(\frac{0.26}{0.74}\right) = \beta_0$$

Statistical Inference in Logistic Regression

The **uncertainty of the estimates** $\hat{\beta}_0$ and $\hat{\beta}_1$ can be quantified and used to calculate both confidence intervals and hypothesis tests.

The estimate for the standard errors of these estimates, likelihood-based, is based on a quantity called Fisher's Information (beyond the scope of this class), which is related to the curvature of the log-likelihood function.

Due to the nature of the underlying Bernoulli distribution, if you estimate the underlying proportion p_i, you get the variance for free! Because of this, the inferences will be based on the normal approximation (and not t-distribution based). Use statsmodels to perform inferences!

Of course, you could always **bootstrap** the results to perform these inferences as well.

Classification using the Logistic Model

$\hat{p}(y=1) \geq 0.5 \Rightarrow$ classify to $y=1$

$\hat{p}(y=1) < 0.5 \Rightarrow$ classify to $y=0$

Using Logistic Regression for Classification

How can we use a logistic regression model to perform classification?

That is, how can we predict when $Y = 1$ vs. when $Y = 0$?

We mentioned before, we can classify all observations for which $\hat{P}(Y = 1) \geq 0.5$ to be in the group associated with $Y = 1$ and then classify all observations for which $\hat{P}(Y = 0) < 0.5$ to be in the group associated with $Y = 0$.

Using such an approach is called the standard **Bayes classifier**.

← Best Classification Accuracy for this model

The Bayes classifier takes the approach that assigns each observation to the most likely class, given its predictor values.

Using Logistic Regression for Classification

When will this Bayes classifier be a good one? When will it be a poor one?

The Bayes classifier is the one that minimizes the overall classification error rate. That is, it minimizes:

$$\frac{1}{n}\sum_{i}^{n} I\left(y_i \neq \hat{y}_i\right)$$

training set.

Is this a good Loss function to minimize? Why or why not?

The Bayes classifier may be a poor indicator within a group. Think about the Heart Data scatter plot...

Using Logistic Regression for Classification

This has potential to be a good classifier if the predicted probabilities are on both sides of 0 and 1.

How do we extend this classifier if Y has more than two categories?

"plurality" (most common/likely class)

Multiple Logistic Regression

Multiple Logistic Regression

It is simple to illustrate examples in logistic regression when there is just one predictors variable.

But the approach 'easily' generalizes to the situation where there are multiple predictors.

A lot of the same details as linear regression apply to logistic regression. Interactions can be considered. Multicollinearity is a concern. So is overfitting. Etc...

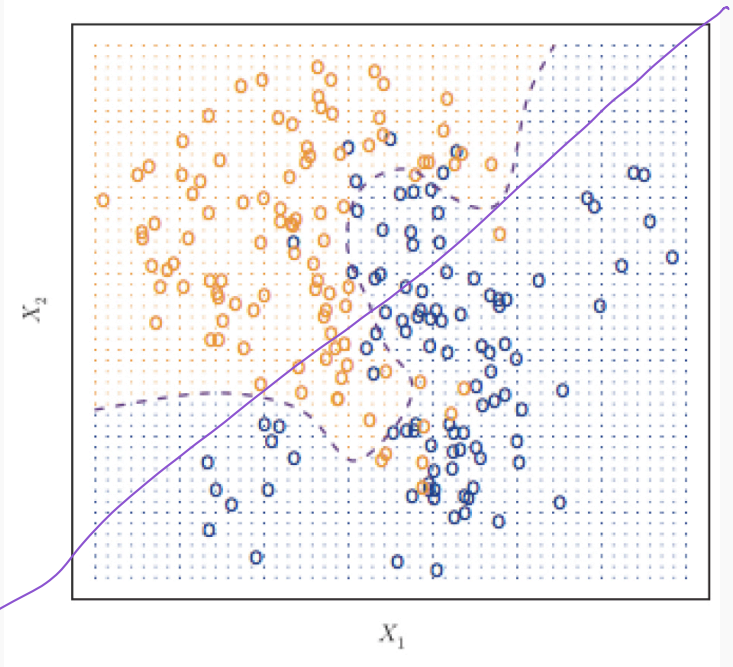
So how do we correct for such problems?

Regularization and checking though train, test, and cross-validation!

We will get into the details of this, along with other extensions of logistic regression, in the next lecture.

Classifier with two predictors

How can we estimate a classifier, based on logistic regression, for the following plot?



Multiple Logistic Regression

Earlier we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model. What was the model statement (in terms of linear predictors)?

$$\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X$$

Multiple logistic regression is a generalization to multiple predictors. More specifically we can define a multiple logistic regression model to predict $P(Y=1)$ as such:

$$\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

Fitting Multiple Logistic Regression

The estimation procedure is identical to that as before for simple logistic regression:

- a likelihood approach is taken, and the function is maximized across all parameters $\beta_0, \beta_1, \ldots, \beta_p$ using an iterative method like Newton-Raphson or Gradient Descent.

The actual fitting of a Multiple Logistic Regression is easy using software (of course there's a python package for that) as the iterative maximization of the likelihood has already been hard coded.

In the `sklearn.linear_model` package, you just have to create your multidimensional design matrix X to be used as predictors in the `LogisticRegression` function.

Interpretation of Multiple Logistic Regression

Interpreting the coefficients in a multiple logistic regression is similar to that of linear regression.

Key: since there are other predictors in the model, the coefficient $\hat{\beta}_j$ is the association between the j^{th} predictor and the response (on log odds scale). But what do we have to say? ... controlling for the effect of other predictors in the model

We are trying to attribute the partial effects of each model controlling for the others (aka, controlling for possible *confounders*).

Interpreting Multiple Logistic Regression: an Example

Let's get back to the Heart Data. We are attempting to predict whether someone has HD based on MaxHR and whether the person is female or male. The simultaneous effect of these two predictors can be brought into one model.

Recall from earlier we had the following estimated models:

$$\log\left(\frac{\widehat{P(Y=1)}}{1-\widehat{P(Y=1)}}\right) = 6.30 - 0.043 \cdot X_{MaxHR}$$

$$\log\left(\frac{\widehat{P(Y=1)}}{1-\widehat{P(Y=1)}}\right) = -1.06 + 1.27 \cdot X_{gender}$$

Interpreting Multiple Logistic Regression: an Example

The results for the multiple logistic regression model are:

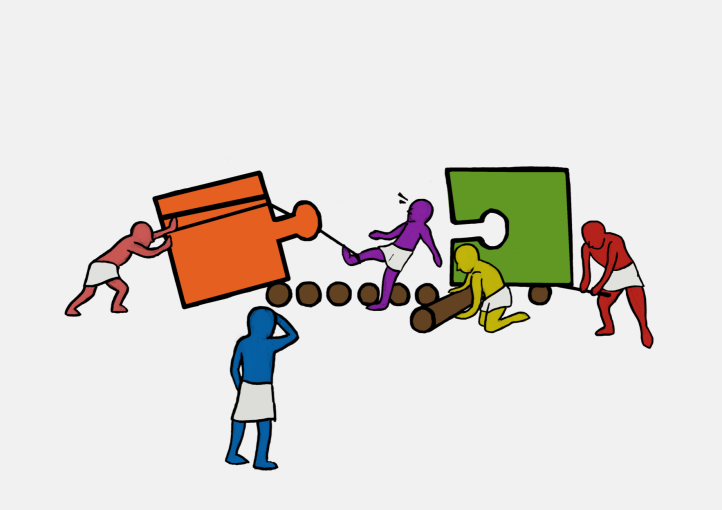
```
data_x = df_heart[['MaxHR','Sex']]
data_y = df_heart['AHD']

logreg = LogisticRegression(C=100000, fit_intercept=True)
logreg.fit(data_x, data_y);


print('Estimated beta1: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1:
 [[-0.04496354  1.40079047]]
Estimated beta0:
 [ 5.58662464]
```

very similar to the separate simple logistic regressions
⇒ very little colinearity

Some questions

1. Estimate the odds ratio of HD comparing men to women using this model.
2. Is there any evidence of multicollinearity in this model?
3. Is there any confounding in this problem?



Exercise Time!

Ex. 2: Simple k-NN and Logistic Regression in `sklearn`
(15+ min)