Linear Classifiers and Logistic Regression

36-462/36-662, Spring 2022

25 January 2022

Housekeeping

- We're back in person next week: Margaret Morrison Hall A14
- Homework 0?
- Homework 1?

In our previous episode

- Regression $\equiv$ predicting a quantitative, numerical variable Y from other variables X (maybe numerical maybe not)
- If we use expected squared error, the optimum prediction rule is to guess $\mu(x) = \mathbb{E}[Y|X = x]$
- If we have to use a *linear* rule, the optimum is to guess $\mathbb{E}[Y] - \frac{\mathrm{Cov}[Y,X]}{\mathrm{Var}[X]}(x - \mathbb{E}[X])$
  - Or similarly for multivariate x
- Ordinary least squares estimates that consistently, pretty generally

Today: classification

- Classification is predicting a binary, categorical Y
- Some linear classification methods
- Some linear methods which also estimate probabilities
- What's a good classifier anyway?

Classification

- We want to predict a binary, categorical Y from other variables X
  - Often convenient to say the two classes are 0 and 1
- A **classifier rule** should map each x to a guess at the right category
  - Multi-class classification: basically similar but more notation, skip for now

Classification accuracy and error rate

- The **error rate** of a classifier rule $m : \mathcal{X} \mapsto \{0, 1\}$ is $\mathbb{P}(Y \neq m(X))$
- The **accuracy** of m is $\mathbb{P}(Y = m(X))$
  - So error rate $= 1-$ accuracy and vice-versa
- Clearly $\mathbb{P}(Y = 1|X = x) \equiv p(x)$ is going to matter here
- The optimal rule $c(x)$ is $c(x) = 1$ if $p(x) \geq 0.5$ and $c(x) = 0$ otherwise
  - You will show this in HW 2

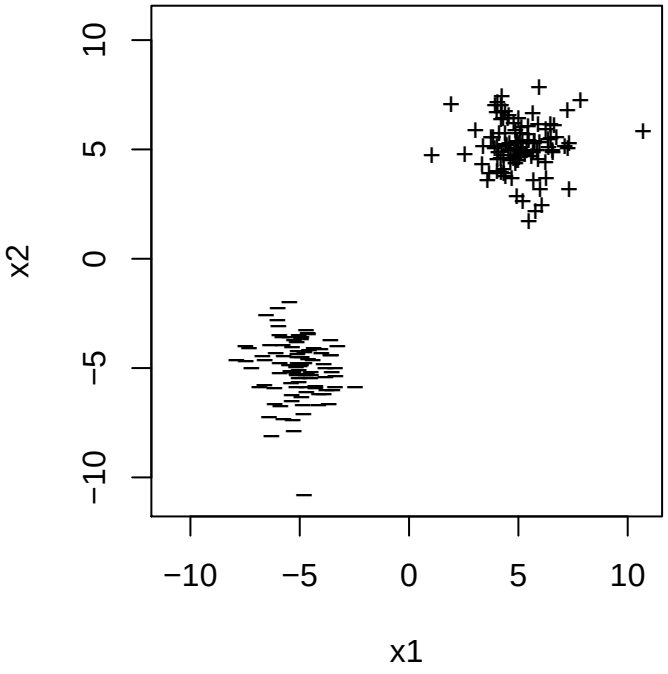
Even the optimal classifier will make mistakes

- Suppose $0 < p(x) < 0.5$
  - Then when $X = x$, the optimal rule makes a mistake with probability $p(x)$ (why?)
- Suppose $0.5 < p(x) < 1$
  - Then when $X = x$, the optimal rule makes a mistake with probability $1 - p(x)$
- The over-all error rate of the optimal rule is $\mathbb{E}\left[\min\left(p(X), 1 - p(X)\right)\right]$
  - This won't, generally, be zero (when will it be?)

A baseline: the constant classifier

- Consider the idiot who ignores X and always predicts the more common class
  - If $\mathbb{P}\left(Y = 1\right) > 0.5$ always predict 1, if $\mathbb{P}\left(Y = 1\right) < 0.5$ always predict 0
- The idiot gets an accuracy of *at least* 50%
  - Potentially much higher if one class is rare
- Always compare your accuracy to the idiot baseline

The prototypical case for the prototype method



The prototype method

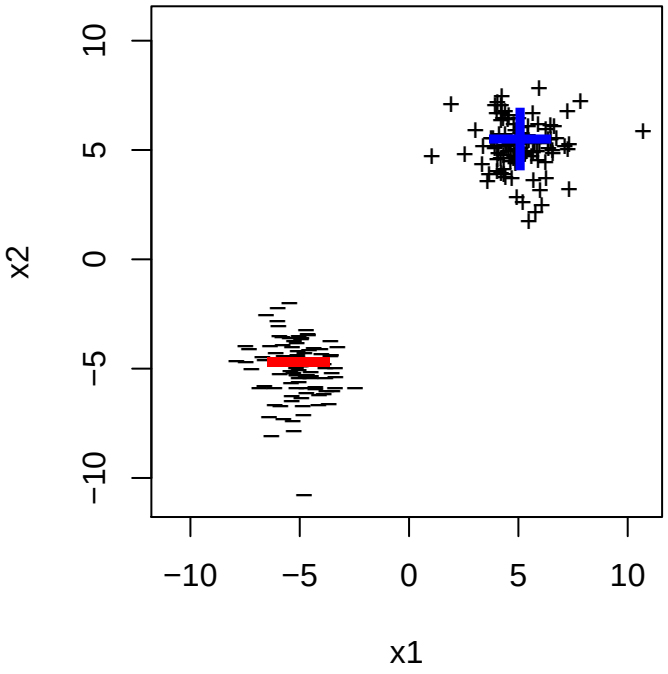
- Find the **prototype** feature vector for each class
  - Prototypes are *usually* the means: for each class c, with n_c data points in the training data,

$$\vec{m}_c = \frac{1}{n_c} \sum_{i:y_i=c} \vec{x}_i$$

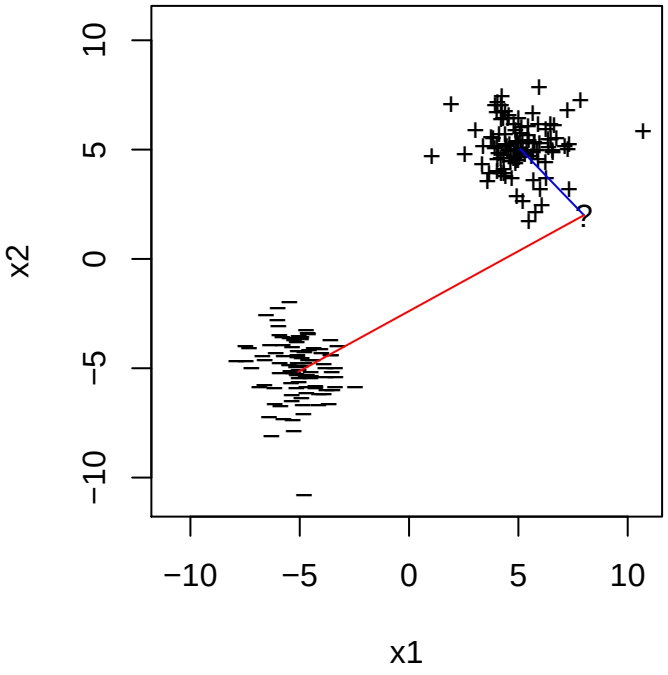
  - Sometimes use medians, trimmed means, etc.
- Classify a new point, $\vec{x}_0$, by seeing which class has the closest prototype to $\vec{x}_0$:

$$\hat{y}(\vec{x}_0) = \operatorname*{argmin}_c \|\vec{x}_0 - \vec{m}_c\|$$

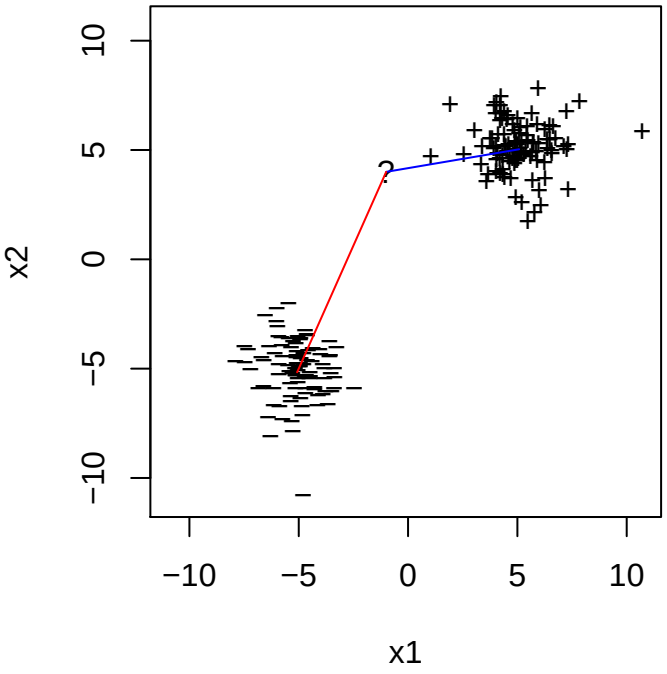
The prototype method in action



The prototype method in action



The prototype method in action

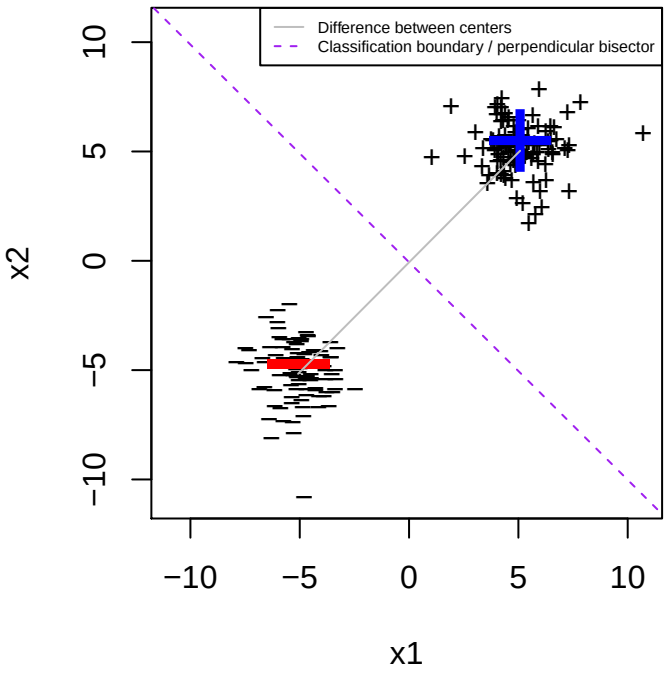


The prototype method...

- will tend to work well when each class forms a clump
- *and* the clumps are widely separated

The boundary

- Where is the **boundary** where our classification switches over?
  - Clearly, it's where the distance to the two class centers is equal.
- We remember how to find that from high school geometry:
  - Draw the line segment connecting the two centers
  - Then draw the **perpendicular bisector** of that line segment



The boundary, with algebra

$$\begin{aligned} \|\vec{x}_0 - \vec{m}_1\| &= \|\vec{x}_0 - \vec{m}_0\| \\ \|\vec{x}_0 - \vec{m}_1\|^2 &= \|\vec{x}_0 - \vec{m}_0\|^2 \\ \|\vec{x}_0\|^2 - 2\vec{x}\cdot\vec{m}_1 + \|\vec{m}_1\|^2 &= \|\vec{x}_0\|^2 - 2\vec{x}\cdot\vec{m}_0 + \|\vec{m}_0\|^2 \\ \|\vec{m}_1\|^2 - \|\vec{m}_0\|^2 &= \vec{x}_0 \cdot 2(\vec{m}_1 - \vec{m}_0) \end{aligned}$$

- So the prototype method is equivalent to the rule

$$\hat{y}(\vec{x}_0) = \begin{cases} 1 & \text{if } \left(\|\vec{m}_0\|^2 - \|\vec{m}_1\|^2\right) + \vec{x}_0 \cdot 2(\vec{m}_1 - \vec{m}_0) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Notice we've got one equation for the boundary, with p unknown coordinates in $\vec{x}_0$, so we get a $(p-1)$-dimensional set of solutions (a line if $p = 2$, a plane if $p = 3$, etc.)
  - The boundary is going to be *perpendicular* to the difference vector $\vec{m}_1 - \vec{m}_0$ (why?)

Linear classifiers

- A **linear classifier** takes the form

$$\hat{y}(\vec{x}_0) = \mathbb{1}\left\{b_0 + \vec{b}\cdot\vec{x}_0 \geq 0\right\}$$

  - $\vec{b}$ is *perpendicular* to the decision boundary (the **normal vector** of the decision boundary), and the **offset** b_0 says how far the decision boundary is from going through the origin
  - (Some people instead write $\mathbb{1}\{b + \vec{w}\cdot\vec{x}_0 \geq 0\}$, etc.)
- Every prototype classifier is a linear classifier, but not vice versa
  - We just saw how to get the offset and the coefficient vector from the locations of the prototypes
- The prototype method doesn't work well when the two classes inter-penetrate or overlap

  - Over-lap pulls the two class centers together
  - We can still try to find a good linear classifier
- Multiple linear classifiers can give the same results
  - $(b_0, \vec{b})$ works just the same as $(ab_0, a\vec{b})$ for any $a > 0$
  - Often (but not always) we standardize so $\|\vec{b}\| = 1$
- Example code:

```
linear.classifier = function(x, coefficients, offset) {
  # The following is actually a (multiple of) the directed distance
  distance.from.plane = function(z) { offset + z %*% coefficients }
  directed.distances = apply(x, 1, directed.distance.from.plane)
  return(ifelse(directed.distances >= 0, 1, 0))
}
```

Margin

- Once we have a boundary, the **margin** of point $\vec{x}_i \equiv$ distance of $\vec{x}_i$ from the boundary
  - We count the distance positively if $\vec{x}_i$ is correctly classified, and *negatively* if $\vec{x}_i$ is mis-classified
- In symbols,
$$\gamma_i(b_0, \vec{b}) = (2y_i - 1)\left(\frac{b_0}{\|\vec{b}\|} + \vec{x}_i \cdot \frac{\vec{b}}{\|\vec{b}\|}\right)$$
  - Stuff inside the parentheses $\equiv$ **directed distance** of $\vec{x}_i$ from the boundary (> 0 when $\vec{x}_i$ is on the positive side)
  - $(2y_i - 1)$ is $+1$ if $y_i = 1$ and -1 if $y_i = 0$
  - So γ_i is, as promised, positive for correctly-classified points and negative for mis-classified points
- The margin of the classifier is the smallest margin of any point:
$$\gamma(b_0, \vec{b}) = \min_{i \in 1:n} \gamma_i(b_0, \vec{b})$$
- $\gamma > 0$ if and only if all the points are correctly classified
- Notice that γ_i is *continuous* in the parameters
  - $\therefore \gamma$ is also continuous in the parameters
  - The *number* of mis-classifications is dis-continuous...

Estimating a linear classifier

- Prototypes
- Maximize the classification accuracy
  - Combinatorial optimization is hard (as we'll go over in a few lectures)
  - Usually many linear boundaries with equal accuracy
- Maximize the margin
  - Continuous optimization is much nicer
  - Prefers saner-looking boundaries
  - Can control out-of-sample error rates in terms of in-sample margin
- "Rosenblatt's Perceptron Rule" (1956): go over the data points one at a time
  - If the current boundary classifies the current point correctly, change nothing and go on to the next point
  - Otherwise, move the boundary in the direction of accommodating the current point, and go on to the next
  - Repeat until nothing changes
  - (Confusion: "perceptron" is also the name for a class of models)

Working probabilities back in

- If a point is very close to the boundary, we shouldn't have much confidence in the classification
- If a point is far away from the boundary, we should have more confidence
  - Unless maybe it's also far away from the training data?
- Let's try to connect probabilities to classifications

First try: linear probability models

- Since $Y = 0$ or $Y = 1$, $\mathbb{E}\left[Y|\vec{X} = \vec{x}\right] = \mathbb{P}\left(Y = 1|\vec{X} = \vec{x}\right)$, so just use linear regression; the decision boundary will be where the predicted response is 0.5
- Pro: requires no knowledge beyond how to type `lm`
- Con: cheerfully predicts probabilities > 1 or < 0
  - You should be embarrassed to say things like "people in X county had a 200% probability of voting for the Republican Party"
- Don't do this
  - Unless you are quite sure your features are *always* going to be in the range where the predicted probability is sensible
  - And you are quite sure that your later users are *never* going to extrapolate outside that range
  - And you are comfortable using a linear regression here (because why, exactly?)

Second try: find a *transformation* of the probability that's linear

- We care about $p(\vec{x}) \equiv \mathbb{P}\left(Y = 1|\vec{X} = \vec{x}\right)$
- Find a transformation of p that's linear in $\vec{x}$
  - $\log p$ won't work (why not?)
- There are many functions which map $[0, 1]$ to $(-\infty, +\infty)$ and are continuous and invertible
  - Inverse of the standard Gaussian CDF ("probit")
  - Inverse of any CDF for a distribution with unbounded range

Think about the likelihood

- We observe data $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots (\vec{x}_n, y_n)$
- Assuming independent responses (given features), the likelihood will be

$$\prod_{i=1}^{n} p(\vec{x}_i)^{y_i}(1 - p(\vec{x}_i))^{1-y_i}$$

  - This is like a Bernoulli or binomial, but now each trial gets its own success probability that's a function of the features
- Re-arrange terms:

$$\prod_{i=1}^{n} (1 - p(\vec{x}_i))\left(\frac{p(\vec{x}_i)}{1 - p(\vec{x}_i)}\right)^{y_i}$$

  - $\frac{p}{1-p}$ is the **odds ratio**; the likelihood only depends on the outcomes (y_i) through the odds ratios
- Take the log:

$$L = \sum_{i=1}^{n} \log\left(1 - p(\vec{x}_i)\right) + y_i \log\left(\frac{p(\vec{x}_i)}{1 - p(\vec{x}_i)}\right)$$

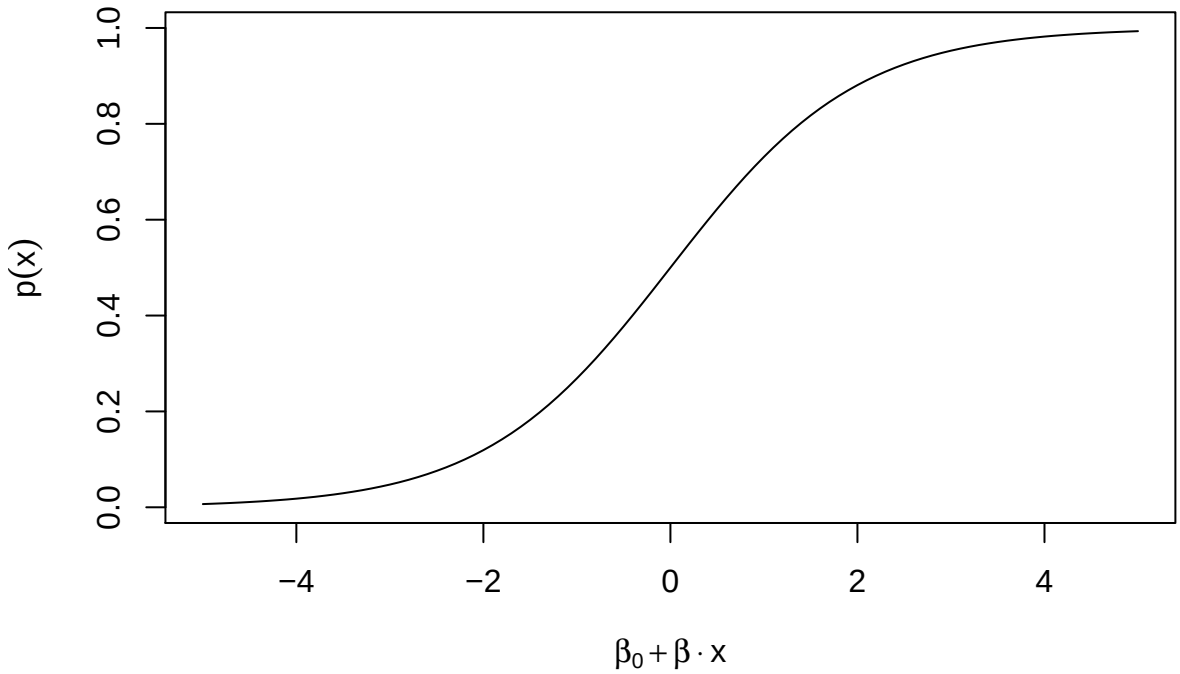
  - The log-likelihood only depends on the outcomes y_i through the **log odds ratio** $\log \frac{p}{1-p}$
  - The log odds ratio maps $[0, 1]$ to $(-\infty, \infty)$ continuously and invertibly

– Any model we use is going to fundamentally involve the log odds ratio, so why don't we make *that* linear in the features?

Logistic regression

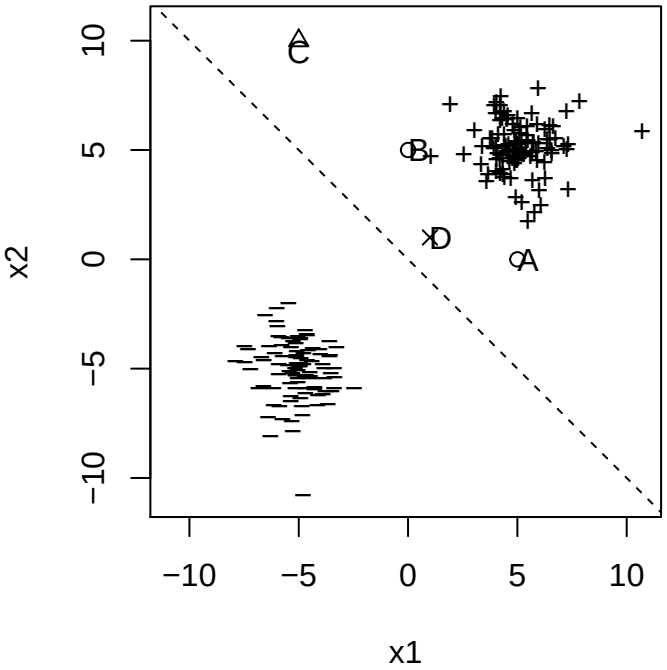
- *Assume* that
$$\log\left(\frac{p(\vec{x})}{1-p(\vec{x}_i)}\right) = \beta_0 + \vec{\beta}\cdot\vec{x}$$
  – Interpretation: unit change in feature x_j adds β_j to the log odds of $Y = 1$ vs. $Y = 0$
- Equivalently, assume that
$$p(\vec{x}) = \frac{e^{\beta_0+\vec{\beta}\cdot\vec{x}}}{1+e^{\beta_0+\vec{\beta}\cdot\vec{x}}} = \frac{1}{1+e^{-(\beta_0+\vec{\beta}\cdot\vec{x})}}$$
  – No easy interpretation of how changing x_j changes the *probability* that $Y = 1$
- **Jargon**: people call $\log p/(1-p)$ the **logit transform** of p (probability $\mapsto$ log-odds), and $e^q/(1+e^q)$ is the **inverse logit transform** (log-odds $\mapsto$ probability)
  – The `faraway` library implements these as the `logit()` and `ilogit()` functions

The logistic curve



- Very large negative values of $\beta_0 + \vec{\beta}\cdot\vec{x}$: probabilities driven to 0
- Exponential take-off as $\beta_0 + \vec{\beta}\cdot\vec{x}$ increases
- Probability 1/2 as $\beta_0 + \vec{\beta}\cdot\vec{x}$ crosses 0
- Very large positive values of $\beta_0 + \vec{\beta}\cdot\vec{x}$: probabilities driven to 1
- "Diminishing returns" or "saturation"
  – If we start from log-odds of 0 (probability 1/2), adding or subtracting 1 to the log-odds changes the probability to 0.73 or 0.27...
  – ... but if we start from a log-odds of 10 (probability 0.9999546), adding or subtracting one from log-odds barely matters (probabilities 0.9999833 or 0.9998766)

Thinking through logistic regression



- Suppose we've estimated a logistic regression here and gotten $\hat{\beta}_0 = 0, \hat{\beta}_1 = 1, \hat{\beta}_2 = 1$
- The classification boundary will be the place where the log-odds $= 0$, so the probability $= 1/2$, or the line $x_2 = -x_1$ (dashed, above)
- Movement *perpendicular* to the boundary changes the log odds
- Movement *parallel* to the boundary does not change the log odds
  - The points A and B will have equal log odds for $Y = 1$ (and those log odds will be > 0)
  - The point C will have slightly lower log odds than either A or B
  - The point D will have *much* lower log odds than either A or B
  - ... even though D is geometrically closer to A and B than C is

How do we estimate logistic regression?

- Maximize the log-likelihood!
- Take derivatives w.r.t. parameters, set equal to zero...
- There is no closed-form solution
  - This is the usual story with maximum likelihood; linear regression with Gaussian noise is kind of weird in that you can write out the maximum explicitly
  - Fortunately, we can optimize numerically
  - We'll come back to that in a week or so
  - WARNING: Maximum likelihood estimation of a logistic regression will *diverge* when you give it linearly-separable data
    - Can you explain why?
    - This is what we have in our running example
    - You should be so lucky as to have this problem in real life
- We'll look at optimization in detail in about a week...

How do we estimate logistic regression?

- In R:

```
glm(y ~ x1 + x2, data = df, family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Call:  glm(formula = y ~ x1 + x2, family = "binomial", data = df)
##
## Coefficients:
## (Intercept)           x1           x2
##       1.462        2.852        3.747
##
## Degrees of Freedom: 199 Total (i.e. Null);  197 Residual
## Null Deviance:       277.3
## Residual Deviance: 2.16e-09  AIC: 6
```

- The `family="binomial"` option tells `glm()` that we're trying to estimate the probability that `y` $== 1$, and the default "link function" is the logistic
  - if you want another transformation of the probability (e.g., "probit"), that's another option
- Notice: warning about not converging due to perfect separation of the classes

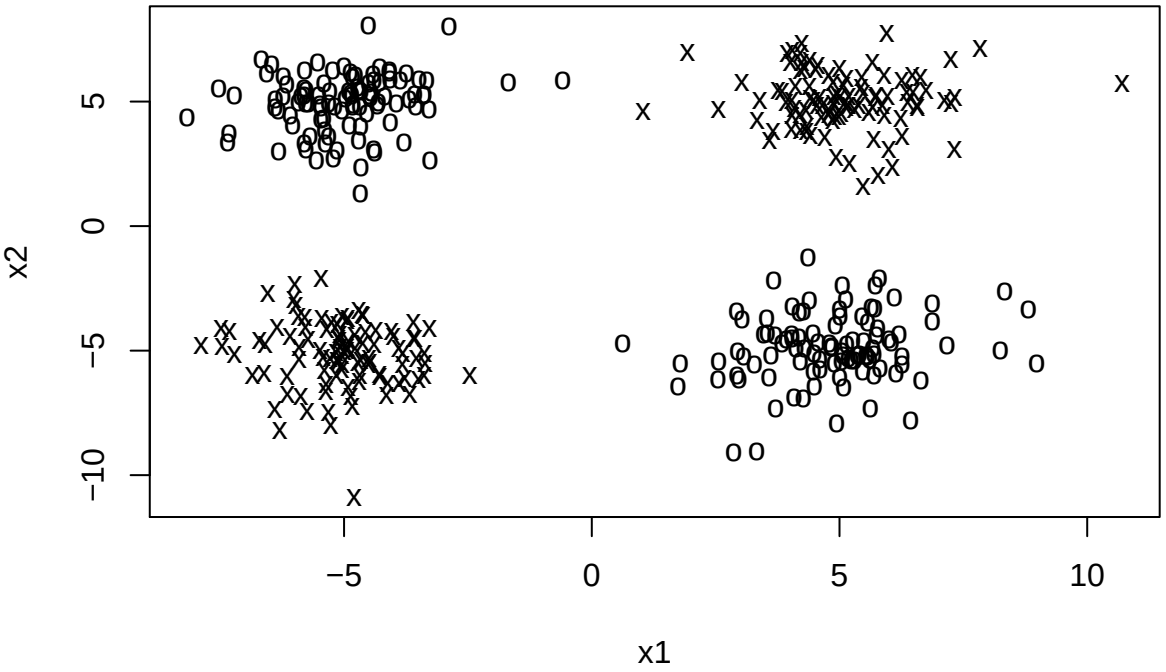
Why logistic regression?

- Tradition / recycling of methods and algorithms from linear regression
  - Tables of coefficients, p-values, confidence intervals...
  - We can do very familiar statistical inference when the model's right
  - Less of a concern for data mining
    - "in the predictive setting, all parameters are nuisance parameters" (Butler 1986, 1)
- It often works pretty well, *especially* if you give it good features
  - ... but lots of things work pretty well, if you give them good features

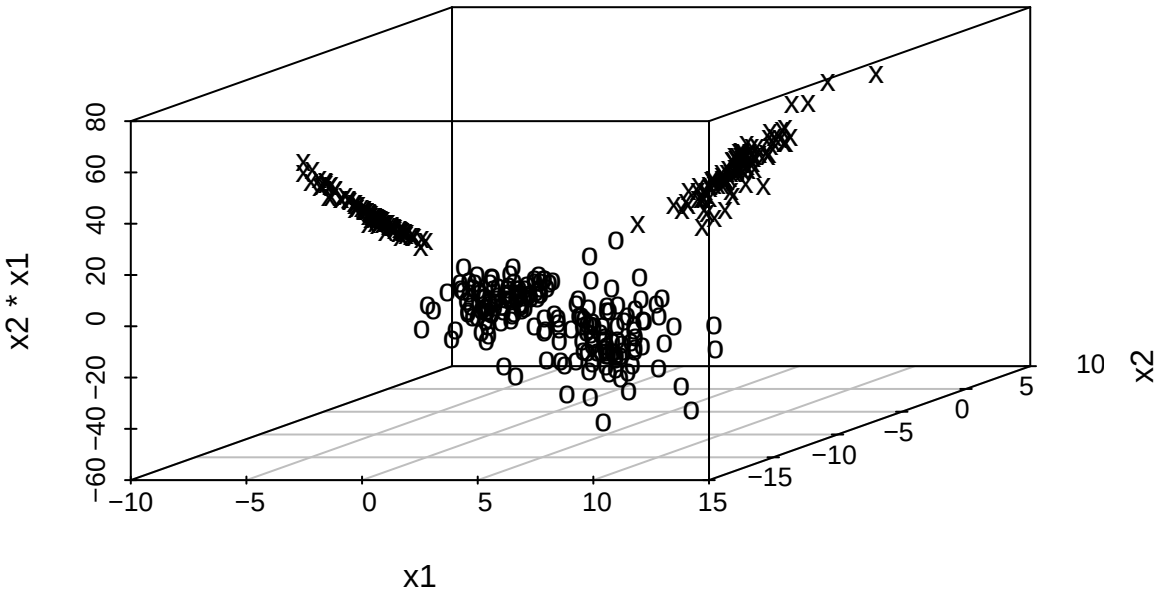
Summing up

- The **prototype method** works well for classification if each class comes from a well-separated clump
- The prototype method is a special case of **linear classification**, where we try to find a linear **boundary** between the classes
- We can often get good performance by looking for classifiers with large **margins**
- **Logistic regression** extends linear classifiers to an actual probability model
  - We can apply any probability threshold we like
  - We can check then model
  - ... all of which may be superfluous if we *just* want to classifty

Going beyond linear classification



- Can't separate the "x" points from the "o" points with a *linear* boundary
- In fact, no *linear* classifier will do much better than chance here
- So should we just give up on linear methods, or is there some way to adapt these ideas? A hint:



Backup: Why "logistic"?

- The function $e^t/(1+e^t)$ is called the **logistic function** or **logistic curve**
- It first showed up in models of population growth against a fixed resource base: starts small, exponential growth, then saturates as the population approaches what resources can support
  - And "logistics" is the art of supplying an army (from the French word *loger*, "to lodge")

- So in one sense this is just re-cycling bits of math because the Ancestors could...
- ... but the log-odds ratio *does* legitimately show up when we try to do maximum likelihood for *any* model with binary responses
  - Making the log-odds linear in $\vec{x}$ is less obvious

References

Butler, Ronald W. 1986. "Predictive Likelihood Inference with Applications." *Journal of the Royal Statistical Society B* 48:1–38. http://www.jstor.org/stable/2345635.