Binary Classification

Wei Xu

(many slides from Greg Durrett and Vivek Srikumar)

This and next Lecture

- Linear classification fundamentals
- Naive Bayes, maximum likelihood in generative models
- Three discriminative models: logistic regression, perceptron, SVM
  - Different motivations but very similar update rules / inference!

Classification

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here: presence or absence of certain words (*great*, *awful*)
- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	...
position 0	position 1	position 2	position 3	position 4	
$f(x)$ = [0	0	1	1	0	...

- Very large vector space (size of vocabulary), sparse features
- Requires *indexing* the features (mapping them to axes)

What are features?

- Don't have to be just *bag-of-words*

$$f(x) = \begin{pmatrix} \text{count(``boring'')} \\ \text{count(``not boring'')} \\ \text{length of document} \\ \text{author of document} \\ \vdots \end{pmatrix}$$

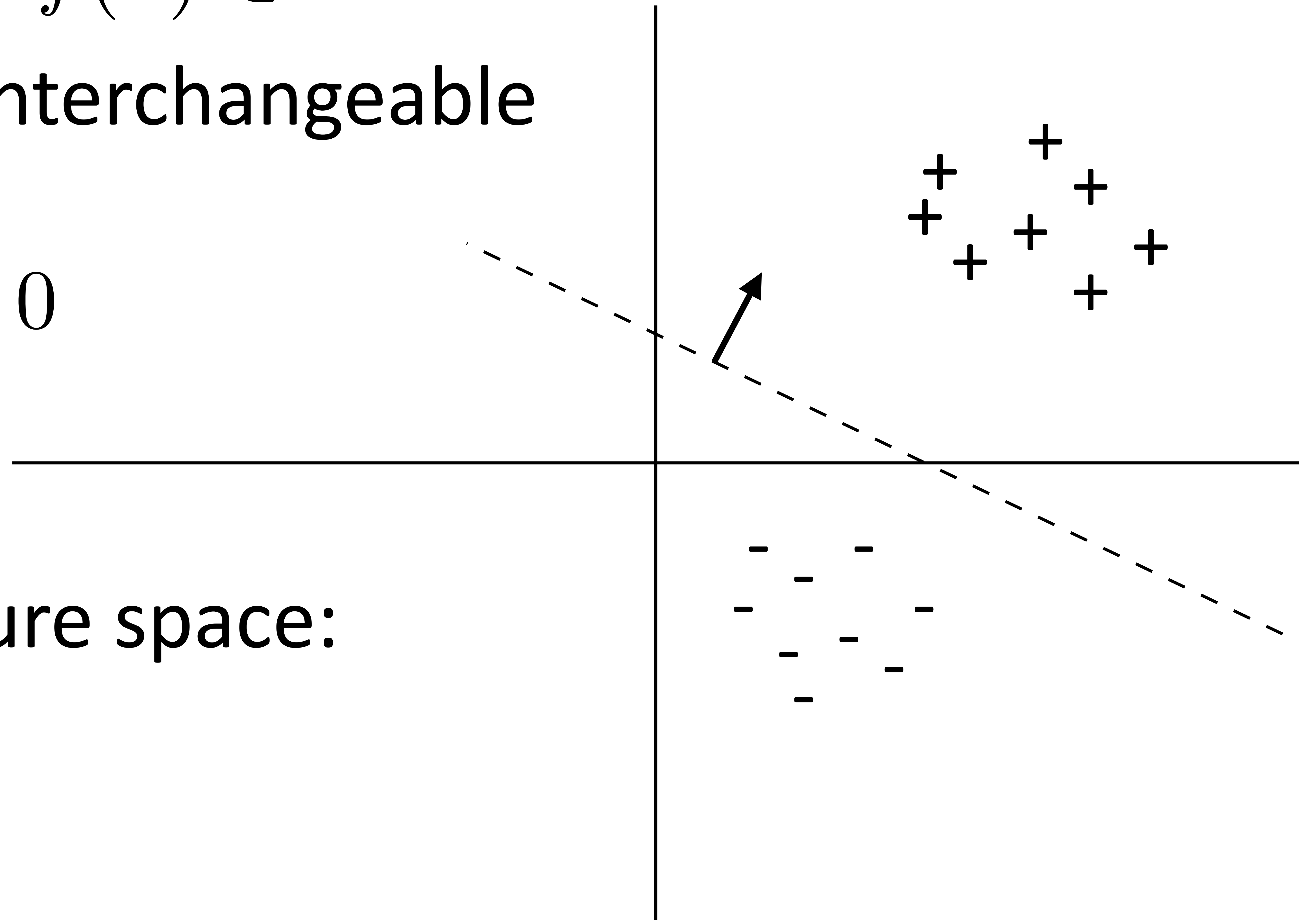
- More sophisticated feature mappings possible (tf-idf), as well as lots of other features: character n-grams, parts of speech, lemmas, …

Classification

- Datapoint x with label $y \in \{0, 1\}$

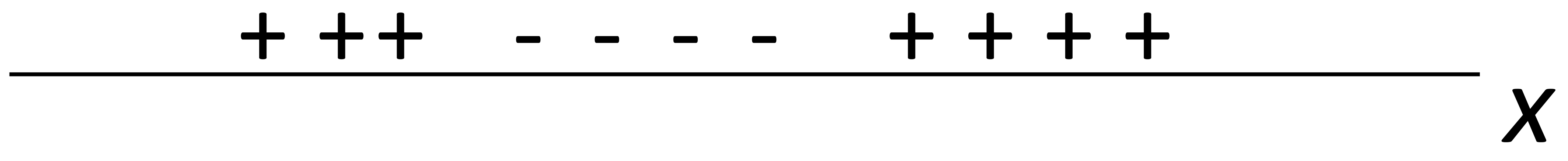
- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
  but in this lecture $f(x)$ and x are interchangeable

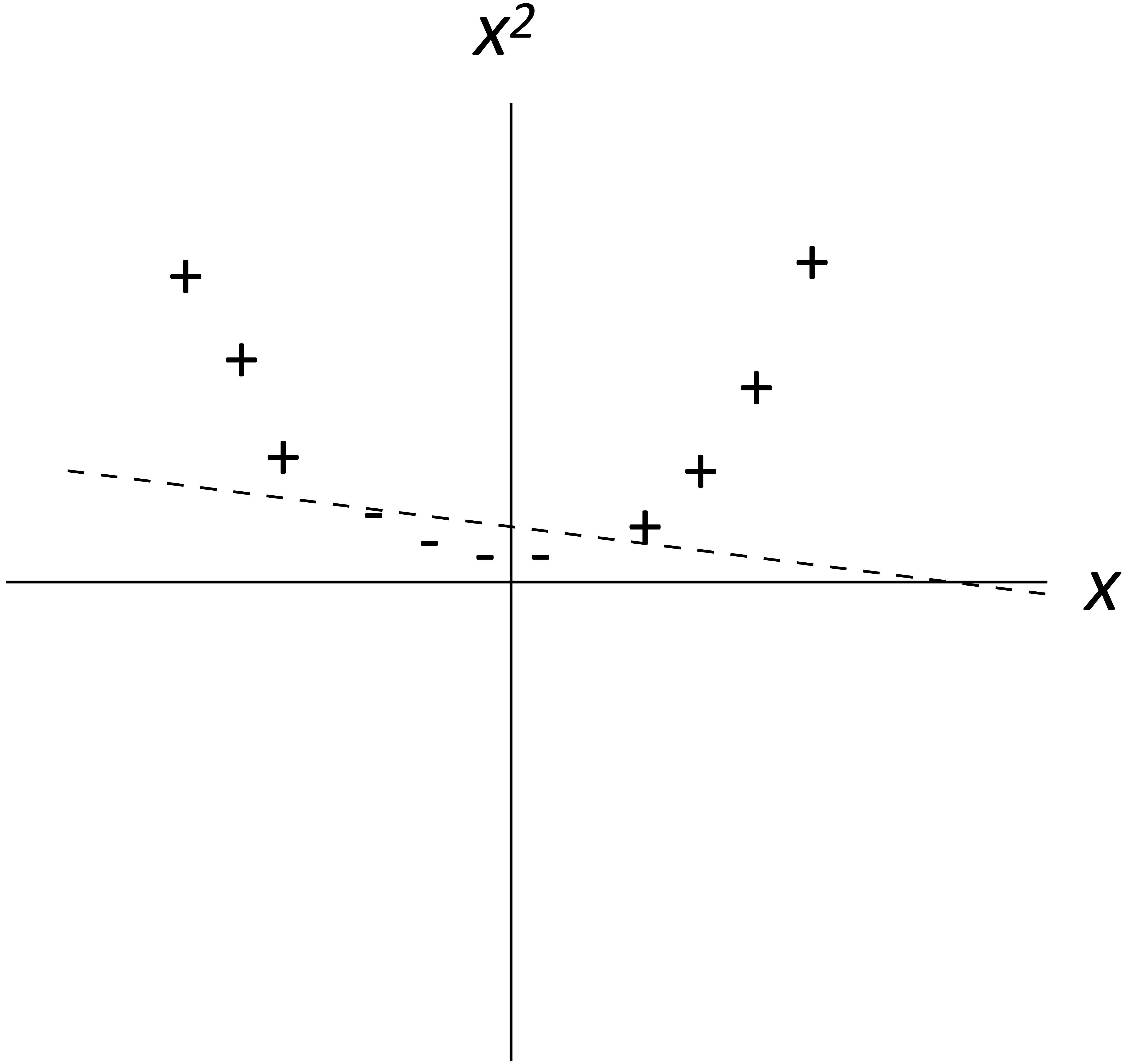
- Linear decision rule: $w^\top f(x) + b > 0$
  $w^\top f(x) > 0$



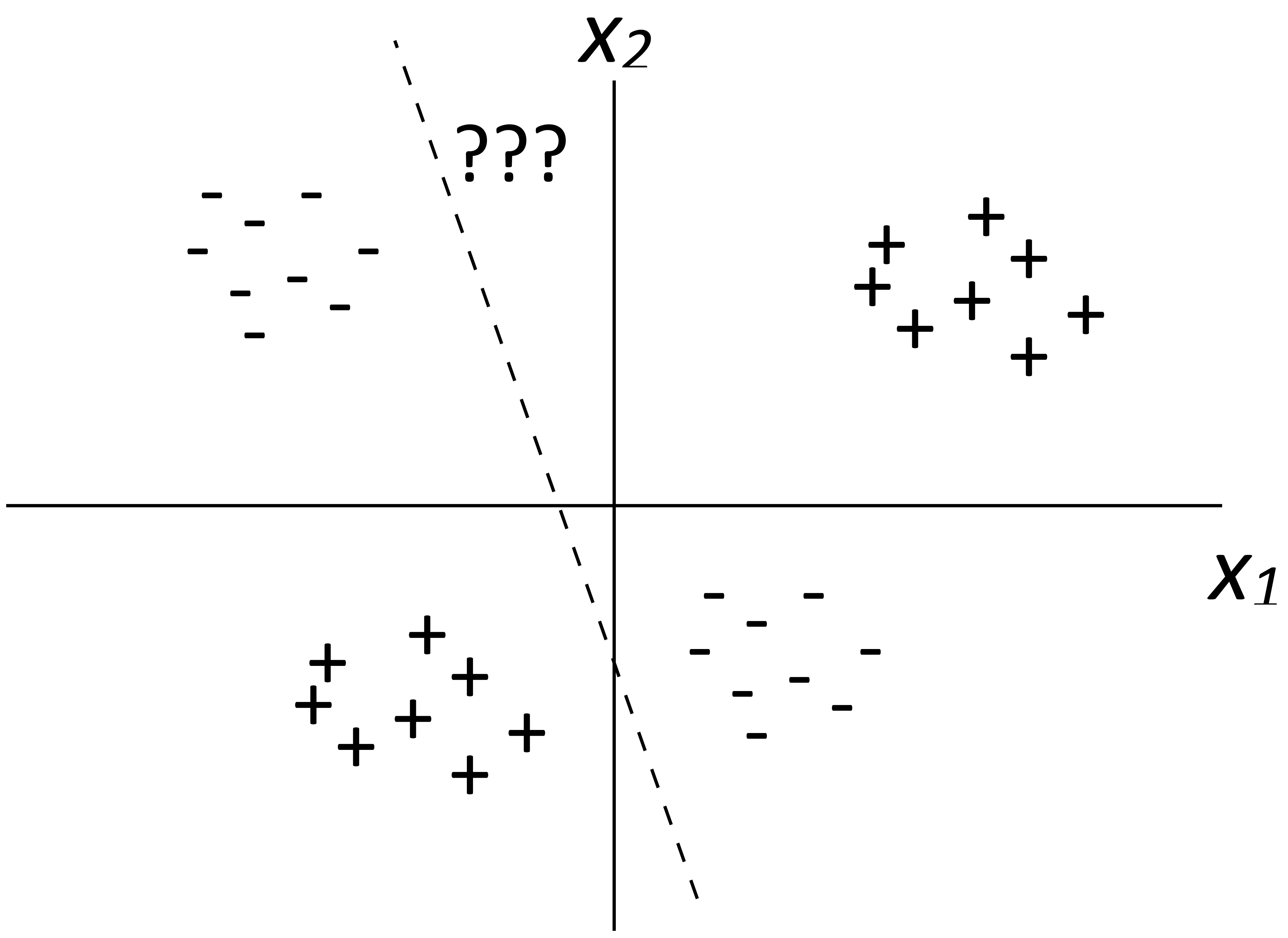
- Can delete bias if we augment feature space:
  $f(x)$ = [0.5, 1.6, 0.3]
  ↓
  [0.5, 1.6, 0.3, **1**]

Linear functions are powerful!

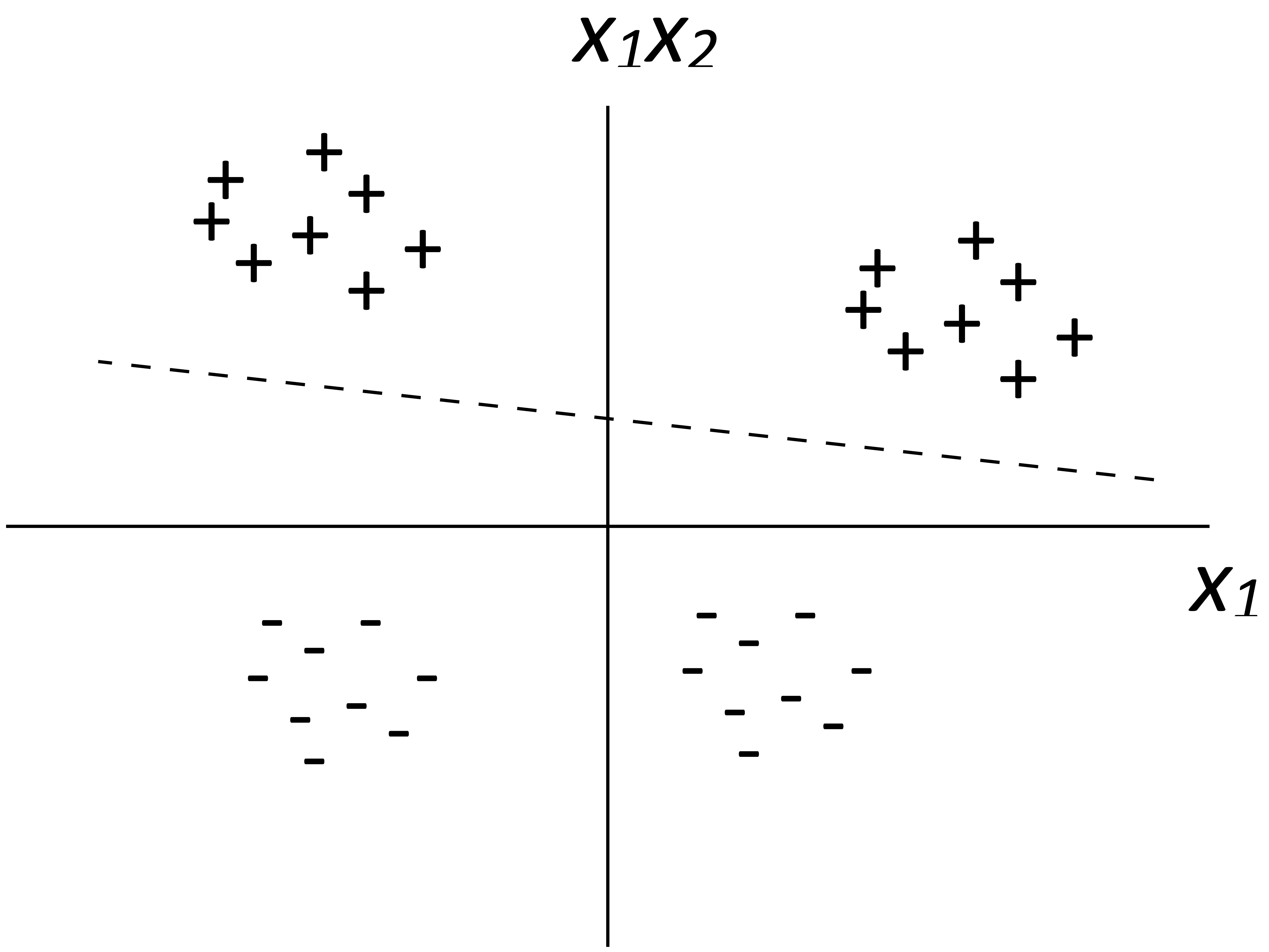




Linear functions are powerful!



$f(x) = [x_1, x_2]$



$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

- "Kernel trick" does this for "free," but is too expensive to use in NLP applications, training is $O(n^2)$ instead of $O(n \cdot (\text{num feats}))$

http://ciml.info/dl/v0_99/ciml-v0_99-ch11.pdf
https://www.quora.com/Why-is-kernelized-SVM-much-slower-than-linear-SVM

Naive Bayes

Naive Bayes

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

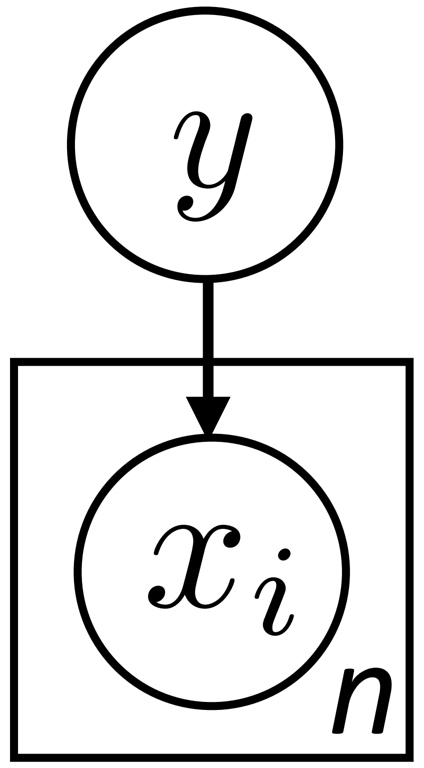
Bayes' Rule

constant: irrelevant
for finding the max

$$\propto P(y)P(x|y)$$

"Naive" assumption:
conditional independence

$$= P(y) \prod_{i=1}^{n} P(x_i|y)$$



$$\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

Why the log?

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)} = P(y)\prod_{i=1}^{n} P(x_i|y)$$

- Multiplying together lots of probabilities
- Probabilities are numbers between 0 and 1

Q: What could go wrong here?

Why the log?

- Problem — floating point underflow

S	exponent	significand
1	11 bits	52 bits

Largest = $1.111\ldots \times 2^{+1023}$

Smallest = $1.000\ldots \times 2^{-1024}$

- Solution: working with probabilities in log space

x	log(x)
0.0000001	-16.118095651
0.000001	-13.815511
0.00001	-11.512925
0.0001	-9.210340
0.001	-6.907755
0.01	-4.605170
0.1	-2.302585

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j)\right]$$

data points (*j*) features (*i*) *i*th feature of *j*th example

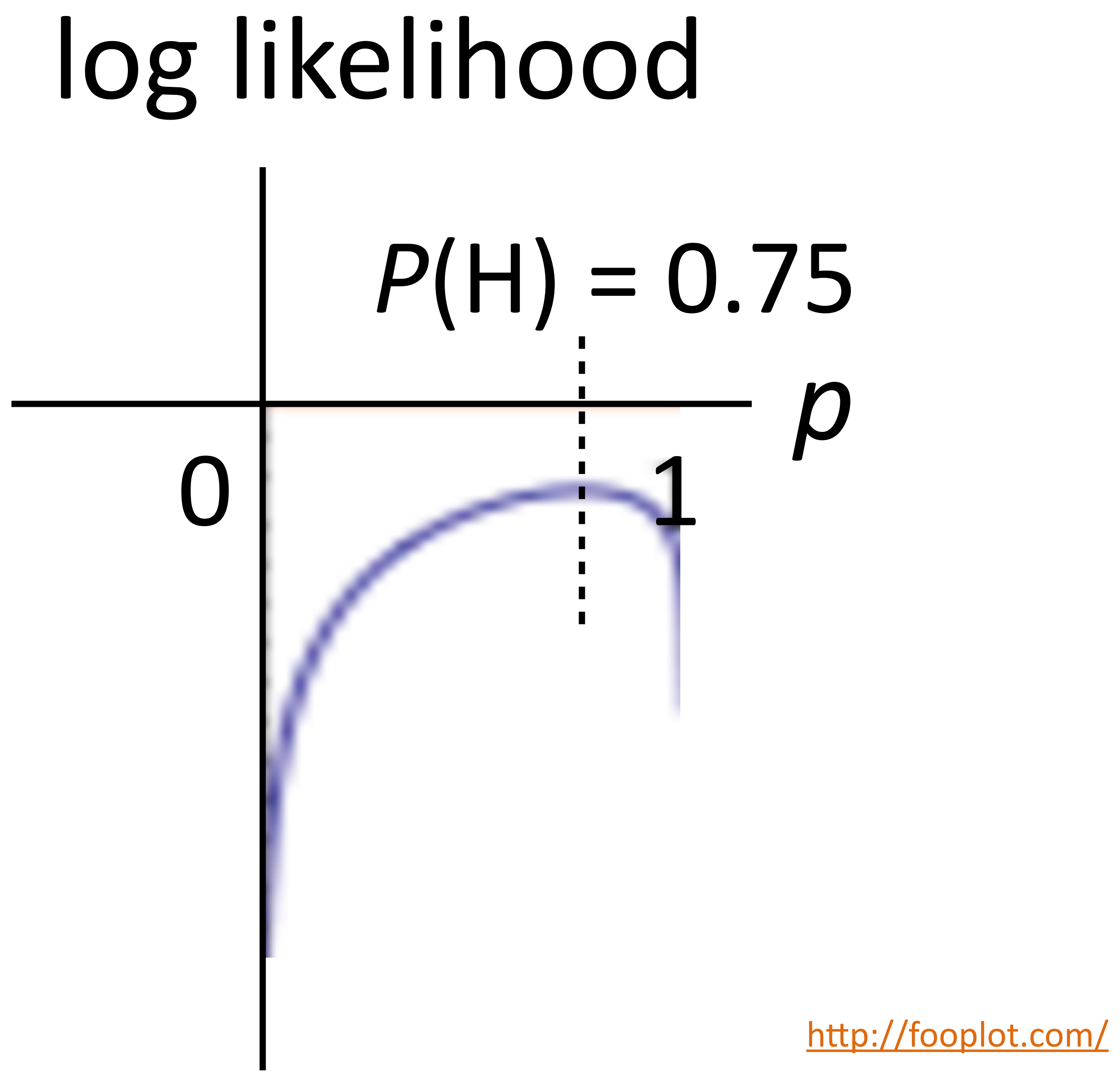
Maximum Likelihood Estimation

- Imagine a coin flip which is heads with probability *p*

- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$

- Easier: maximize *log* likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3 \log p + \log(1-p)$$



Maximum Likelihood Estimation

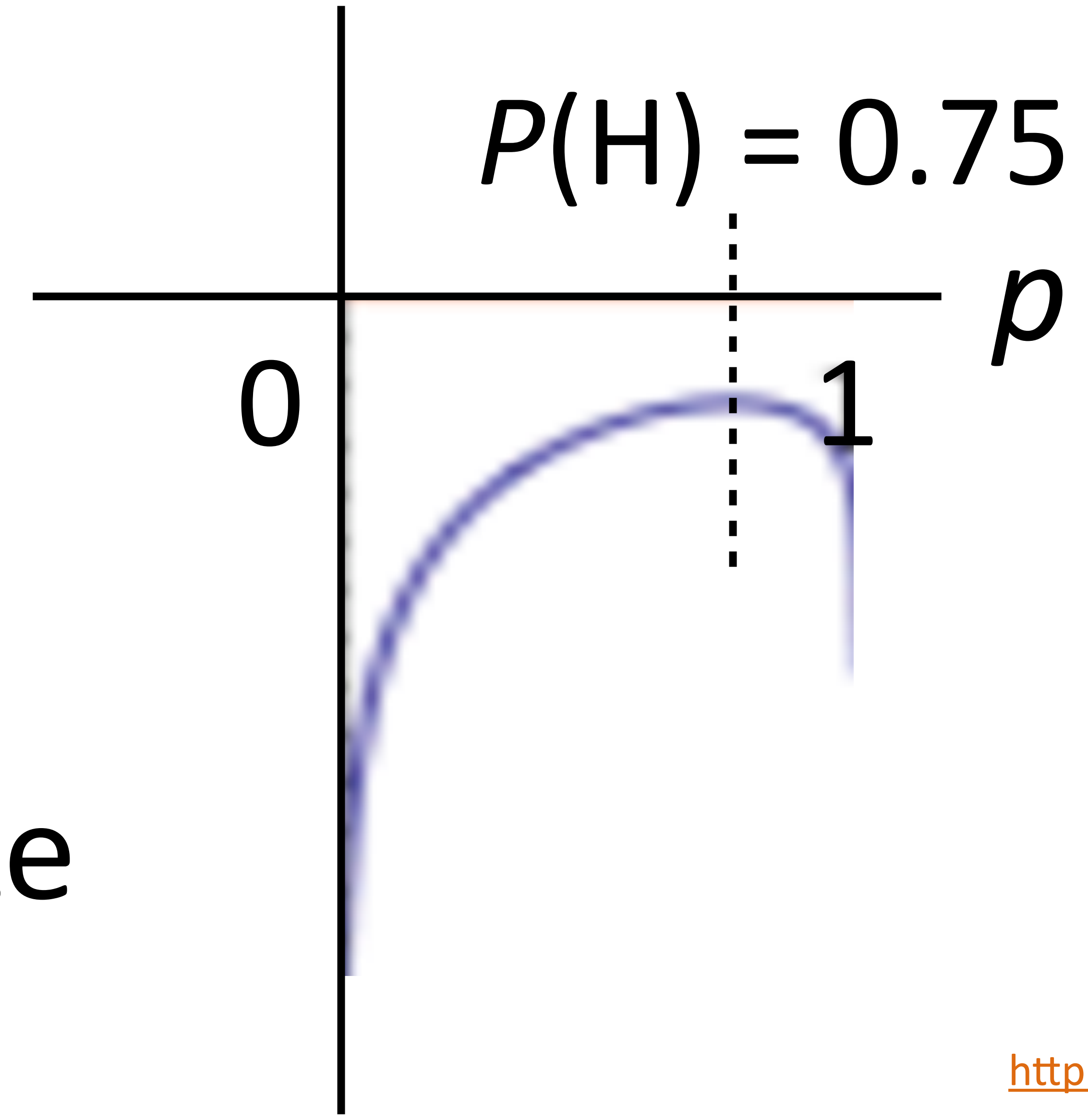
- Imagine a coin flip which is heads with probability *p*

- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$

- Easier: maximize *log* likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3\log p + \log(1-p)$$

log likelihood



- Maximum likelihood parameters for binomial/ multinomial = read counts off of the data + normalize

http://fooplot.com/