# Introduction to Generative Modeling

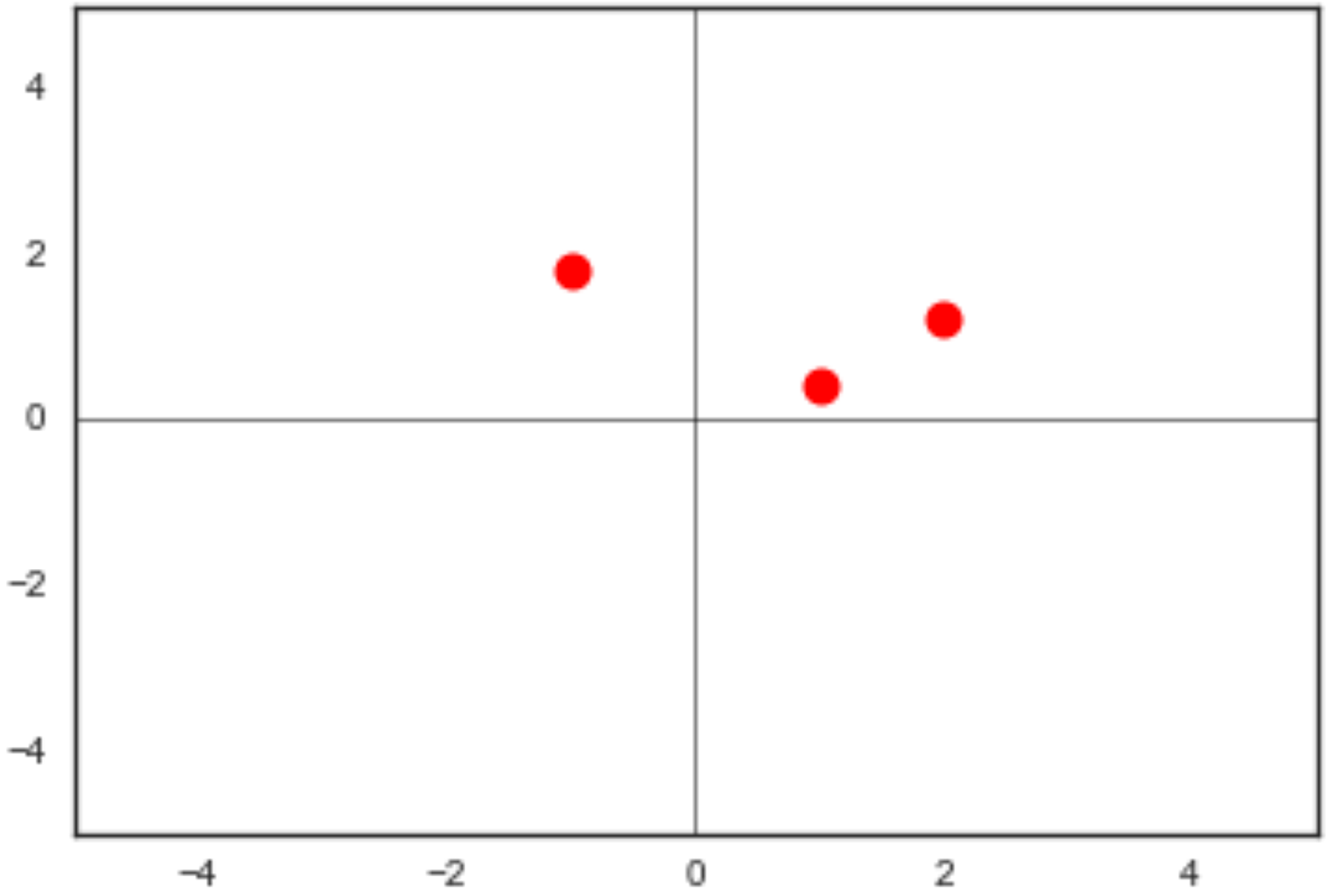
Brooks Paige


# What is a model?

- Generative view: a *model* is a simulator
- What is “understanding”?

- If you are able to simulate a system, then you understand the system

# Generative model for curve fitting




Draw a line “through” these points

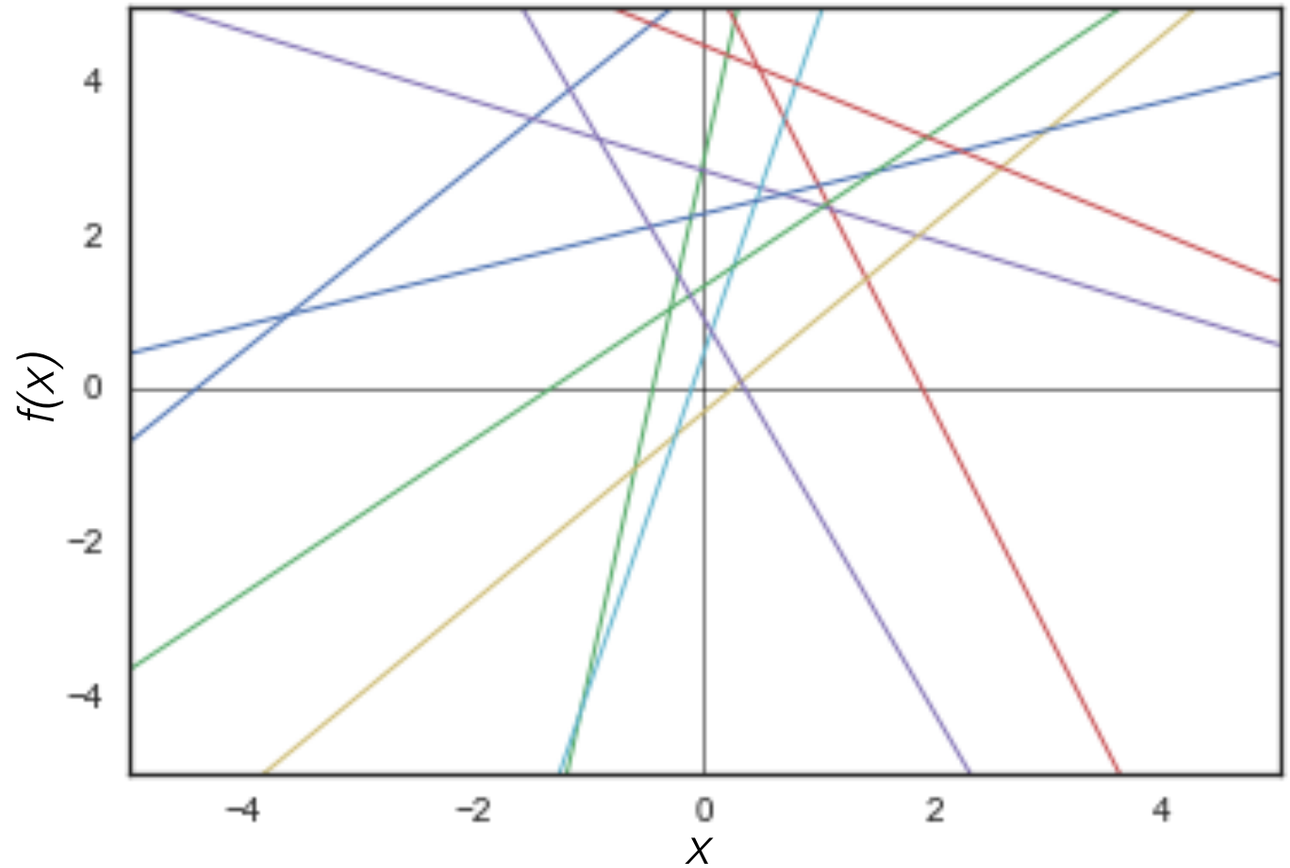
# Generative model for curve fitting

- Goal of curve fitting / regression tasks: estimate a function *f(x)* with *y = f(x) + noise*

- Question: how do we "generate" a line?

- *y = a + bx + noise*

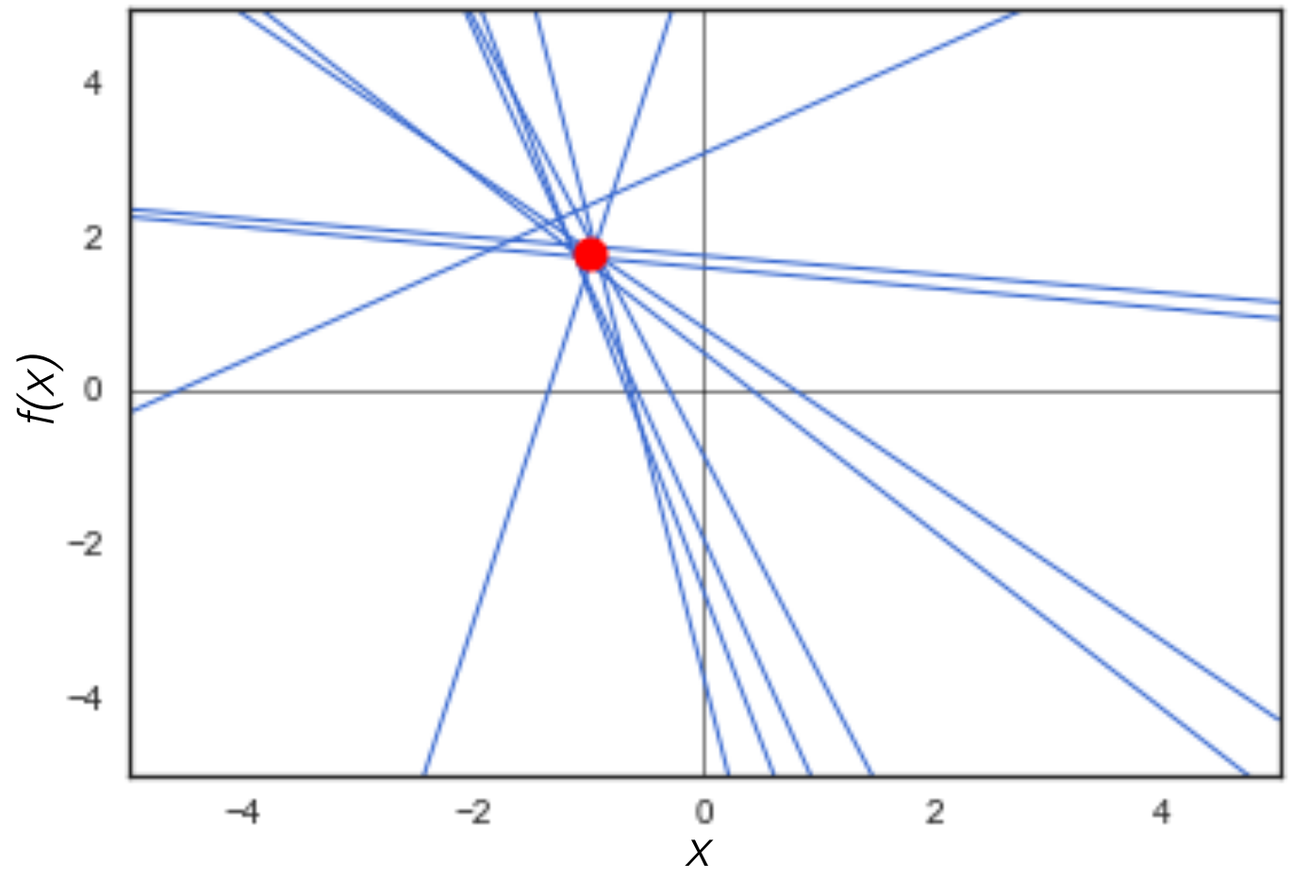
    1. choose an intercept
    2. choose a slope
    3. add in noise



# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*

- "Good" sets of latent variables are those which well simulate the data



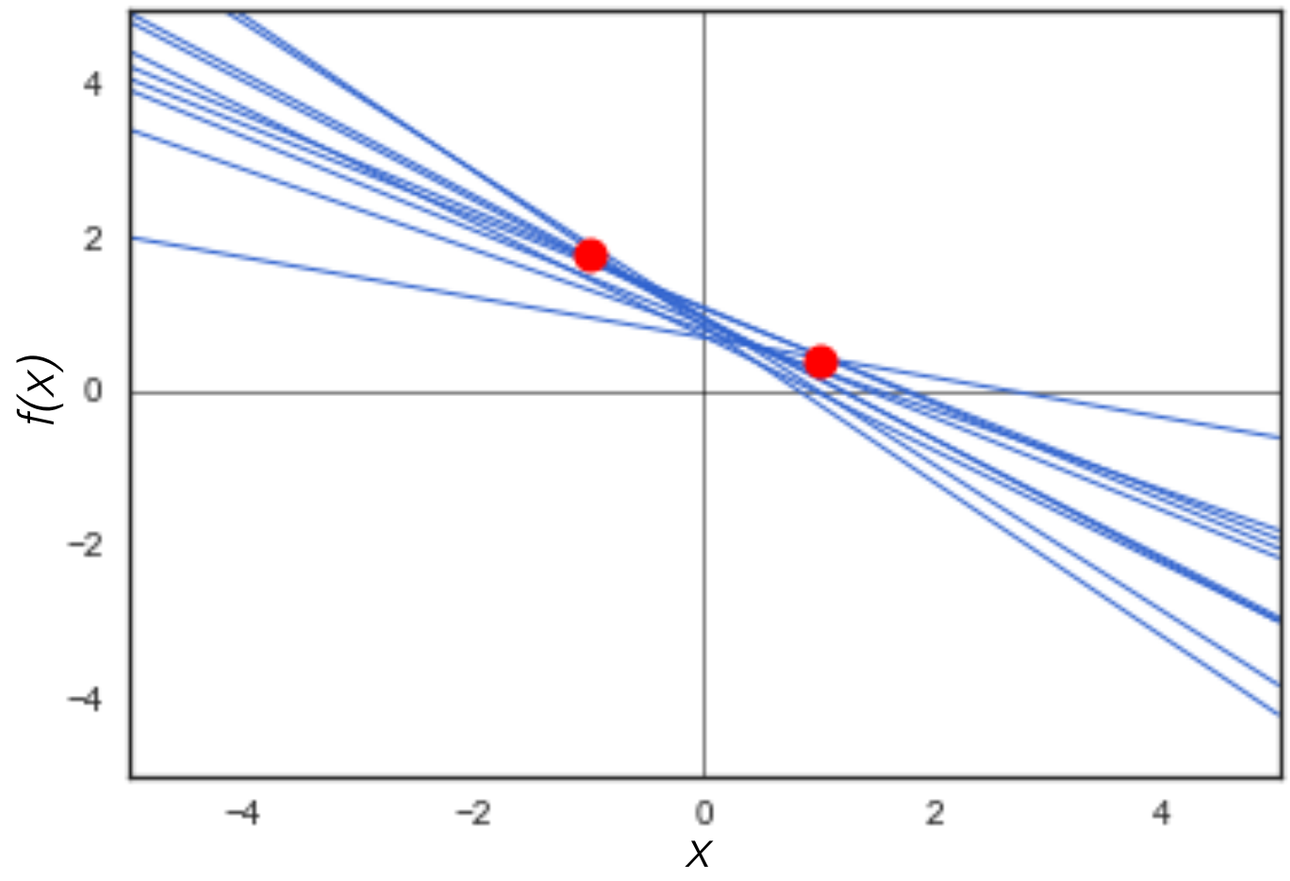

- *y = a + bx + noise*

  1. choose an intercept
  2. choose a slope
  3. add in noise

# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*

- "Good" sets of latent variables are those which well simulate the data



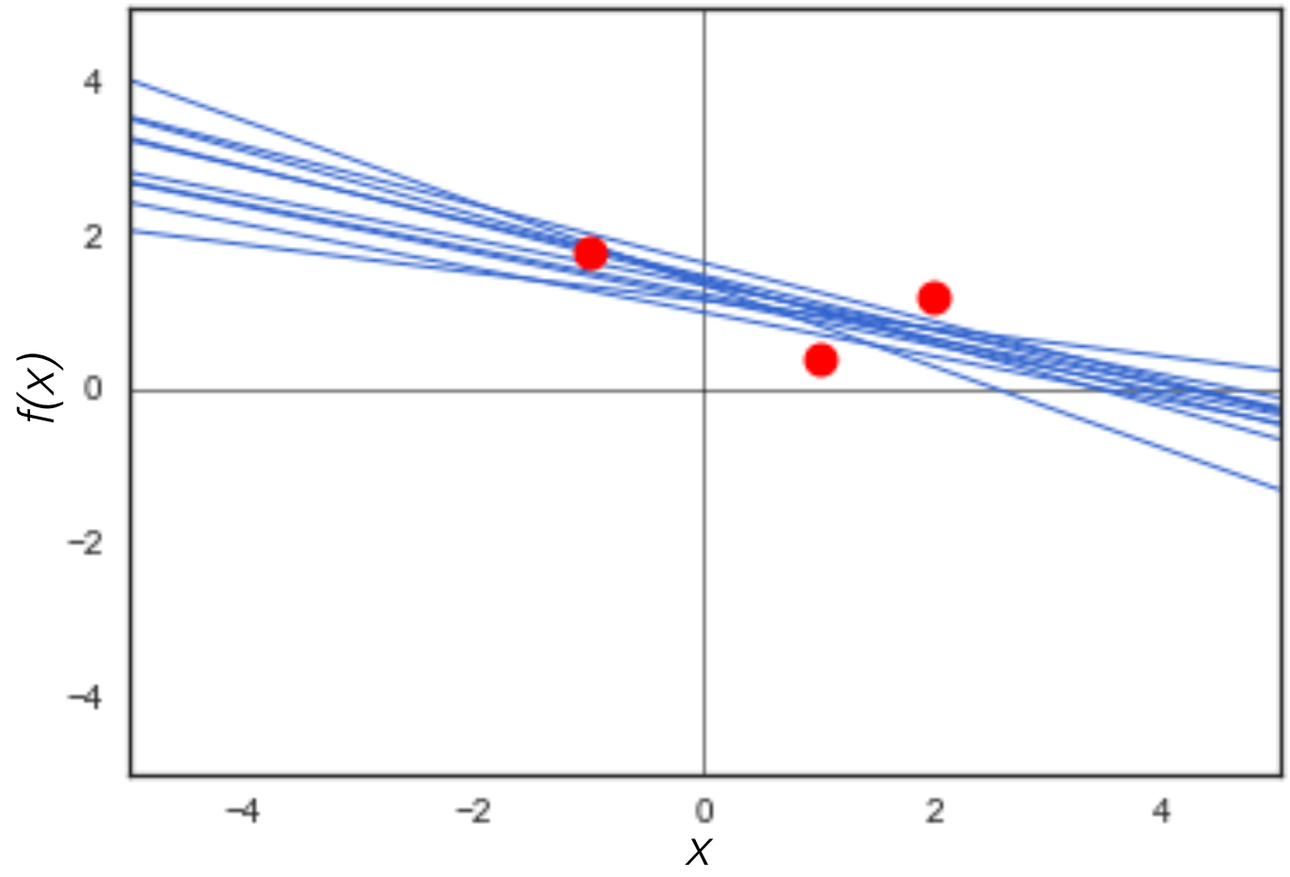

- *y = a + bx + noise*

  1. choose an intercept
  2. choose a slope
  3. add in noise

# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*

- "Good" sets of latent variables are those which well simulate the data

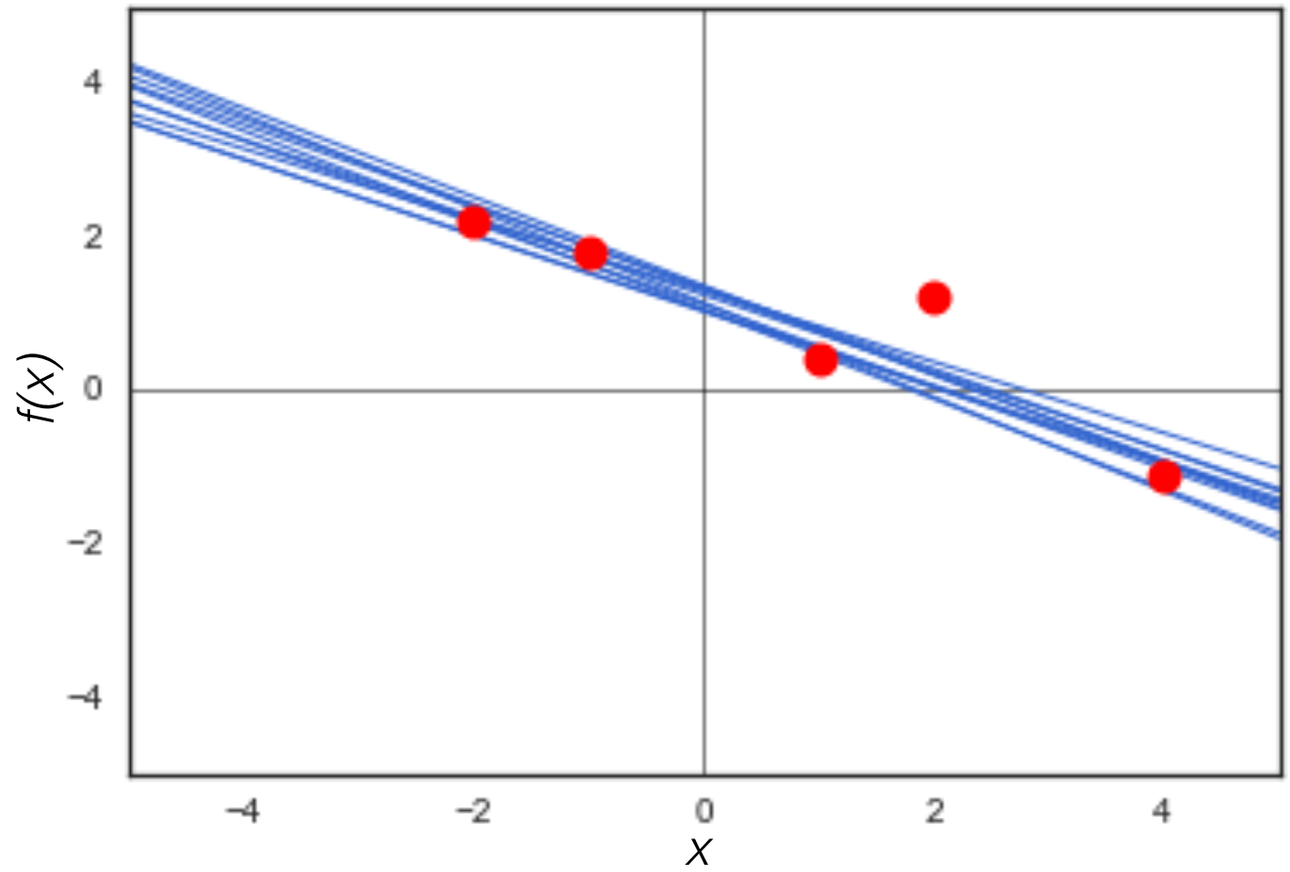



- *y = a + bx + noise*

  1. choose an intercept
  2. choose a slope
  3. add in noise

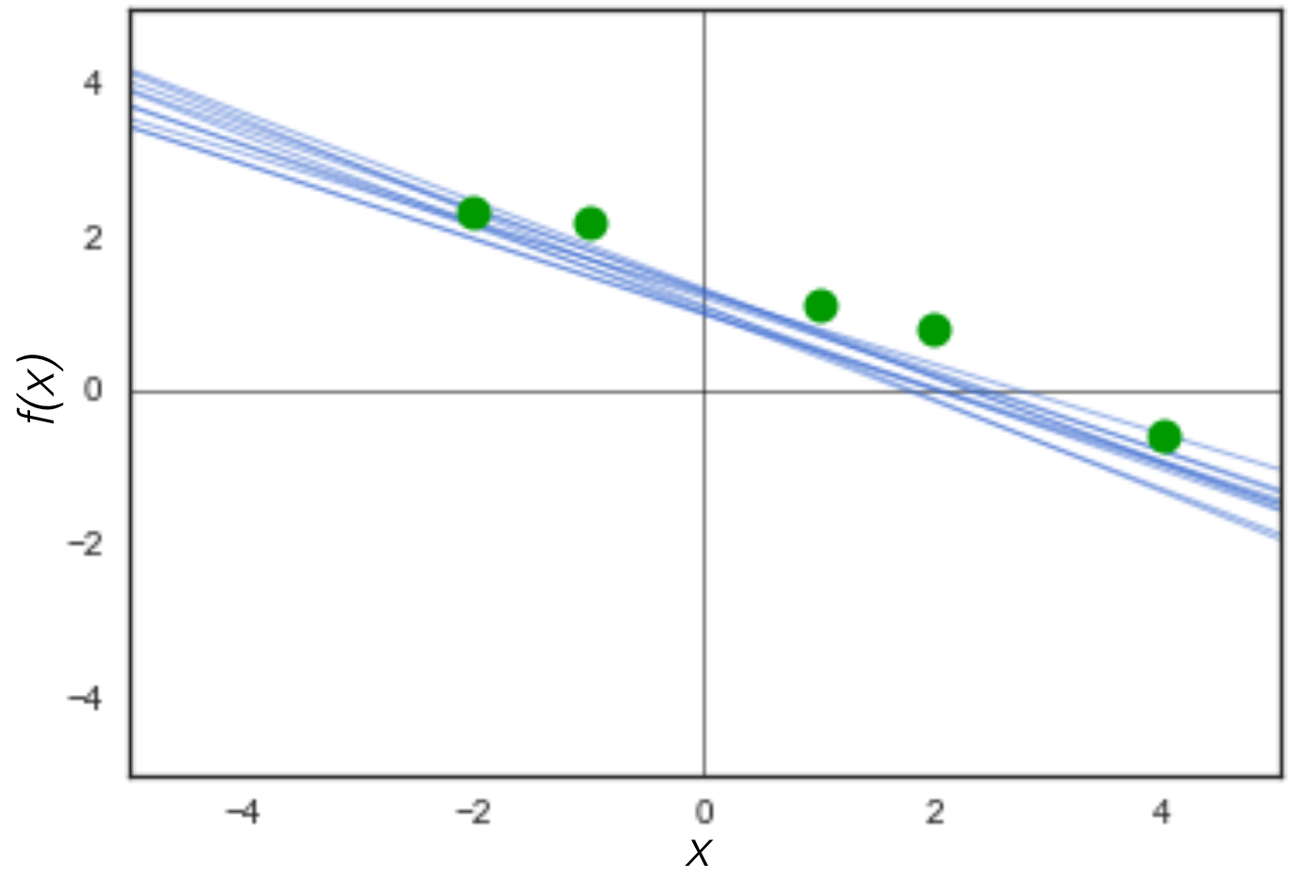
# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*
- "Good" sets of latent variables are those which well simulate the data
- *y = a + bx + noise*
  1. choose an intercept
  2. choose a slope
  3. add in noise



# Generative model for curve fitting

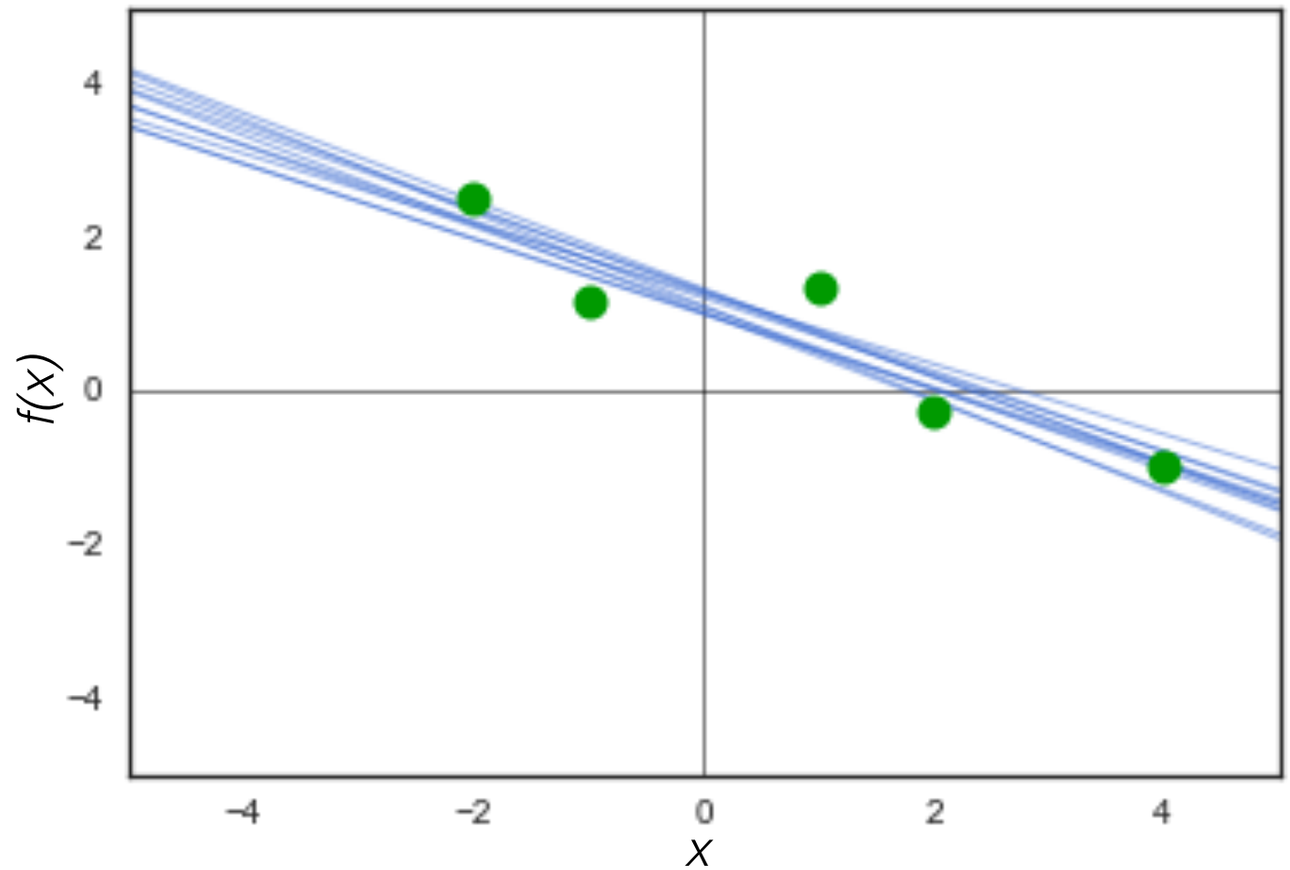
- Define a joint probability distribution *p(a, b, noise)*
- "Good" sets of latent variables are those which well simulate the data
- *y = a + bx + noise*
  1. choose an intercept
  2. choose a slope
  3. add in noise



# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*

- "Good" sets of latent variables are those which well simulate the data

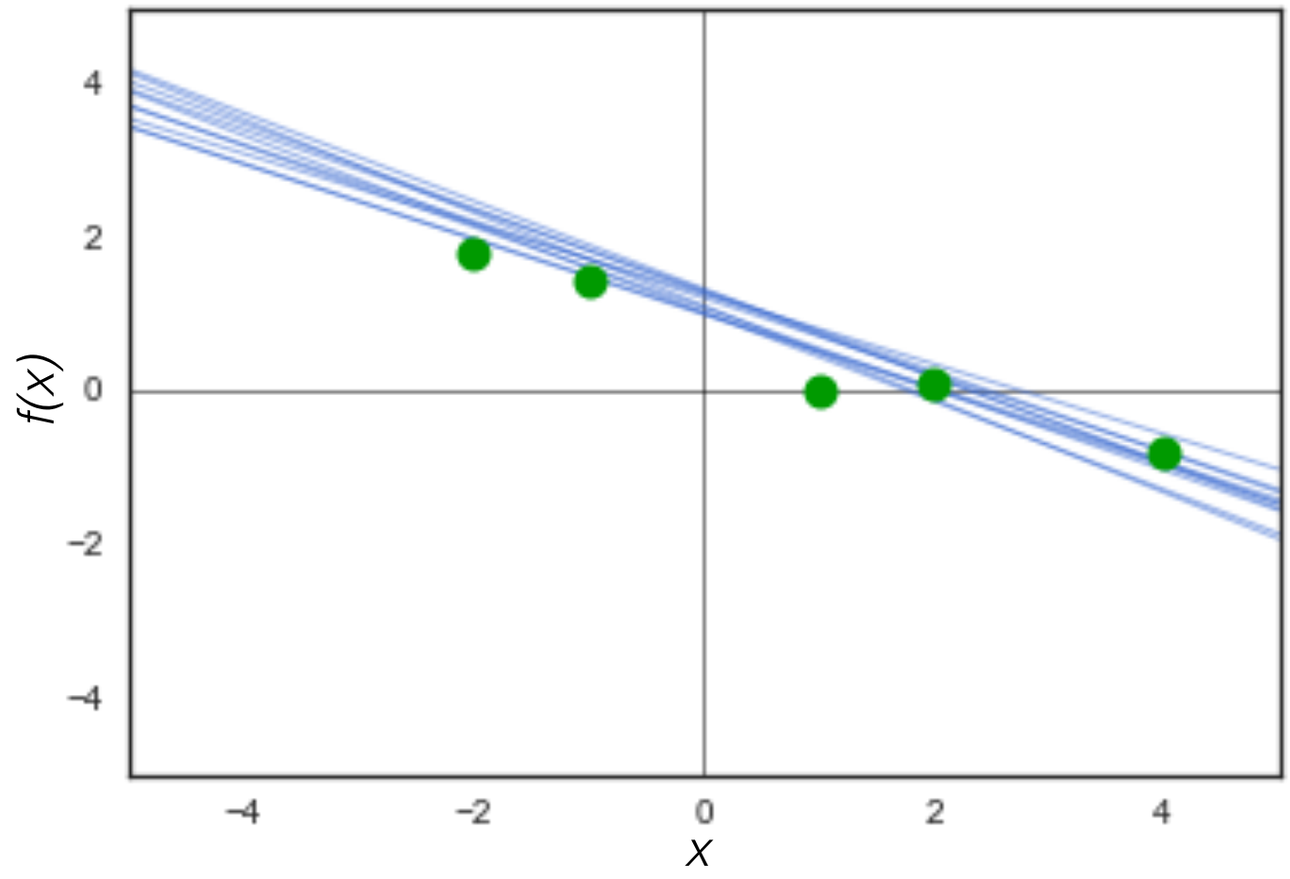



- *y = a + bx + noise*

  1. choose an intercept
  2. choose a slope
  3. add in noise

# Generative model for curve fitting

- Define a joint probability distribution *p(a, b, noise)*
- "Good" sets of latent variables are those which well simulate the data
- *y = a + bx + noise*
  1. choose an intercept
  2. choose a slope
  3. add in noise



# Generative models and Bayes' rule

- A generative model specifies how to simulate what we see (i.e. the data), as well as what we *don't* see (*latent variables* or parameters)

- If we have a generative model, we can answer questions (or "test hypotheses") using Bayes' rule:

$$p(hypothesis \mid data) = \frac{p(data \mid hypothesis)\, p(hypothesis)}{p(data)}$$

# Generative model for curve fitting

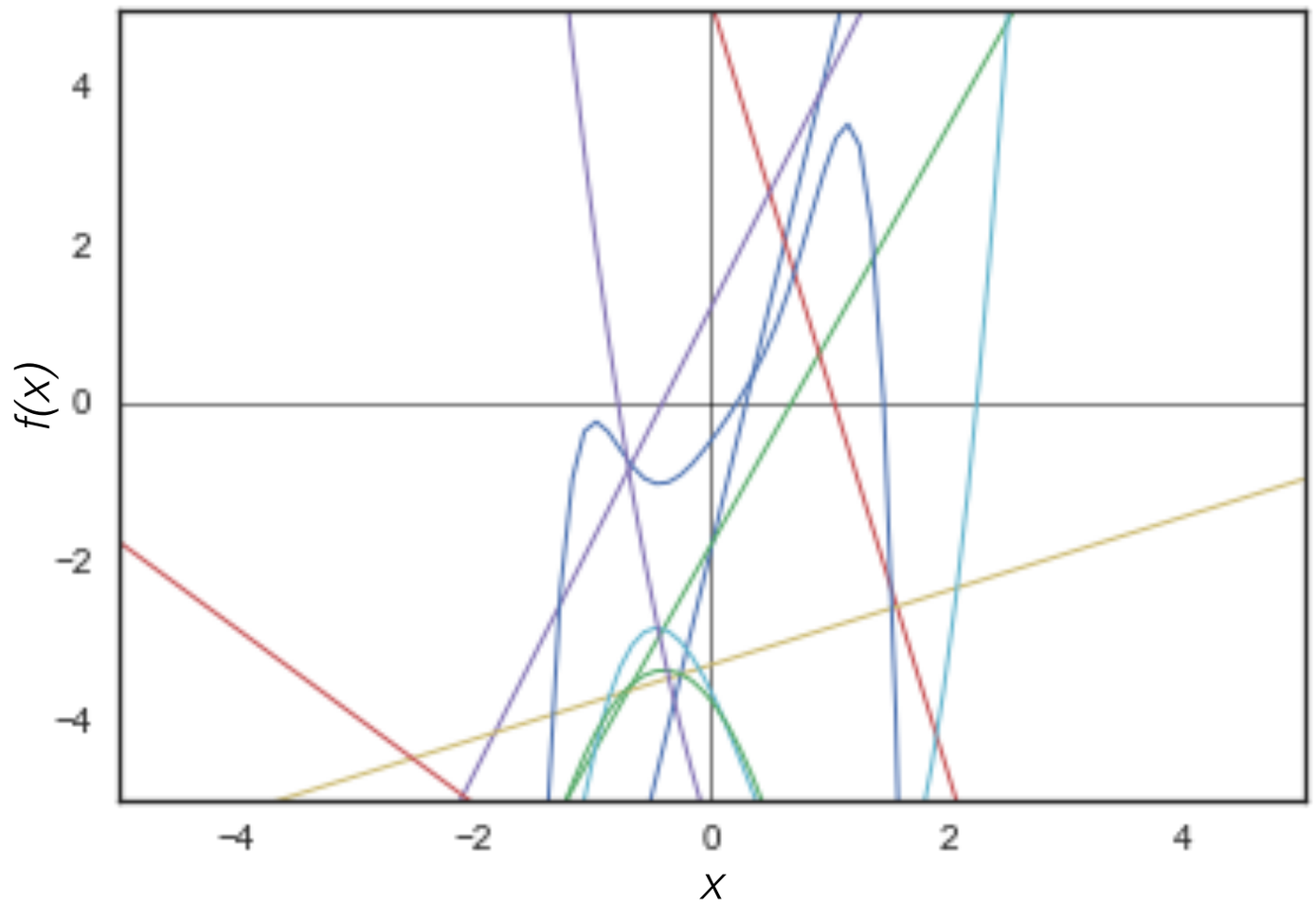
- Our generative model for a line: *p(f)* defined a probability distribution over **linear functions**
- We simulated from the generative model by sampling a slope and an intercept, and then adding noise
- What if we want to sample more interesting functions?

# Generative model for curve fitting

- Polynomial functions: $f(x) = w_0 + \sum_{i=1}^{D} w_i x^i$

- First, choose (sample) degree of polynomial *D* from some distribution over positive integers

- Then, choose (sample) values for each of the *D+1* different coefficients

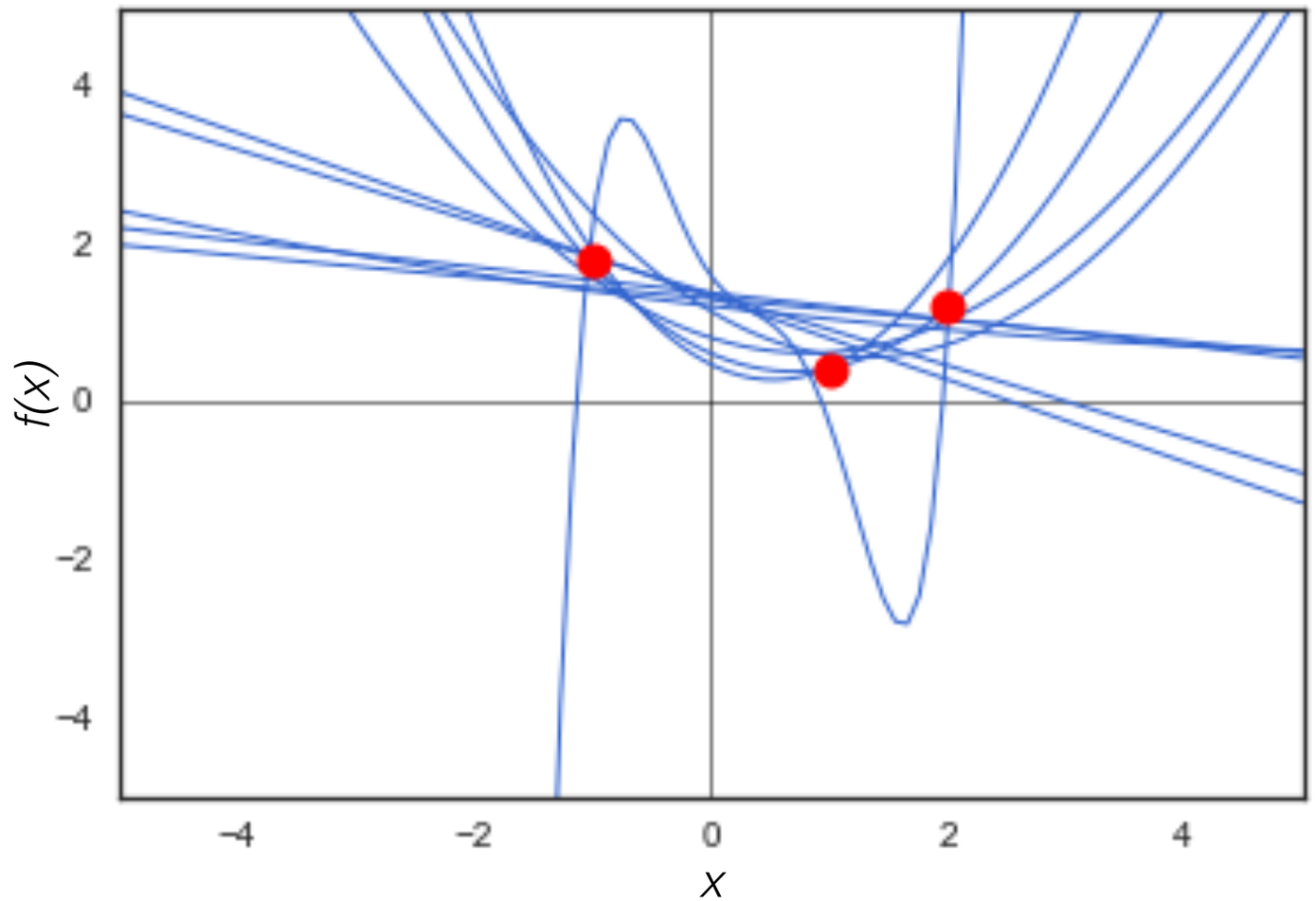


# Generative model for curve fitting

- Polynomial functions: $f(x) = w_0 + \sum_{i=1}^{D} w_i x^i$

- First, choose (sample) degree of polynomial *D* from some distribution over positive integers

- Then, choose (sample) values for each of the *D+1* different coefficients

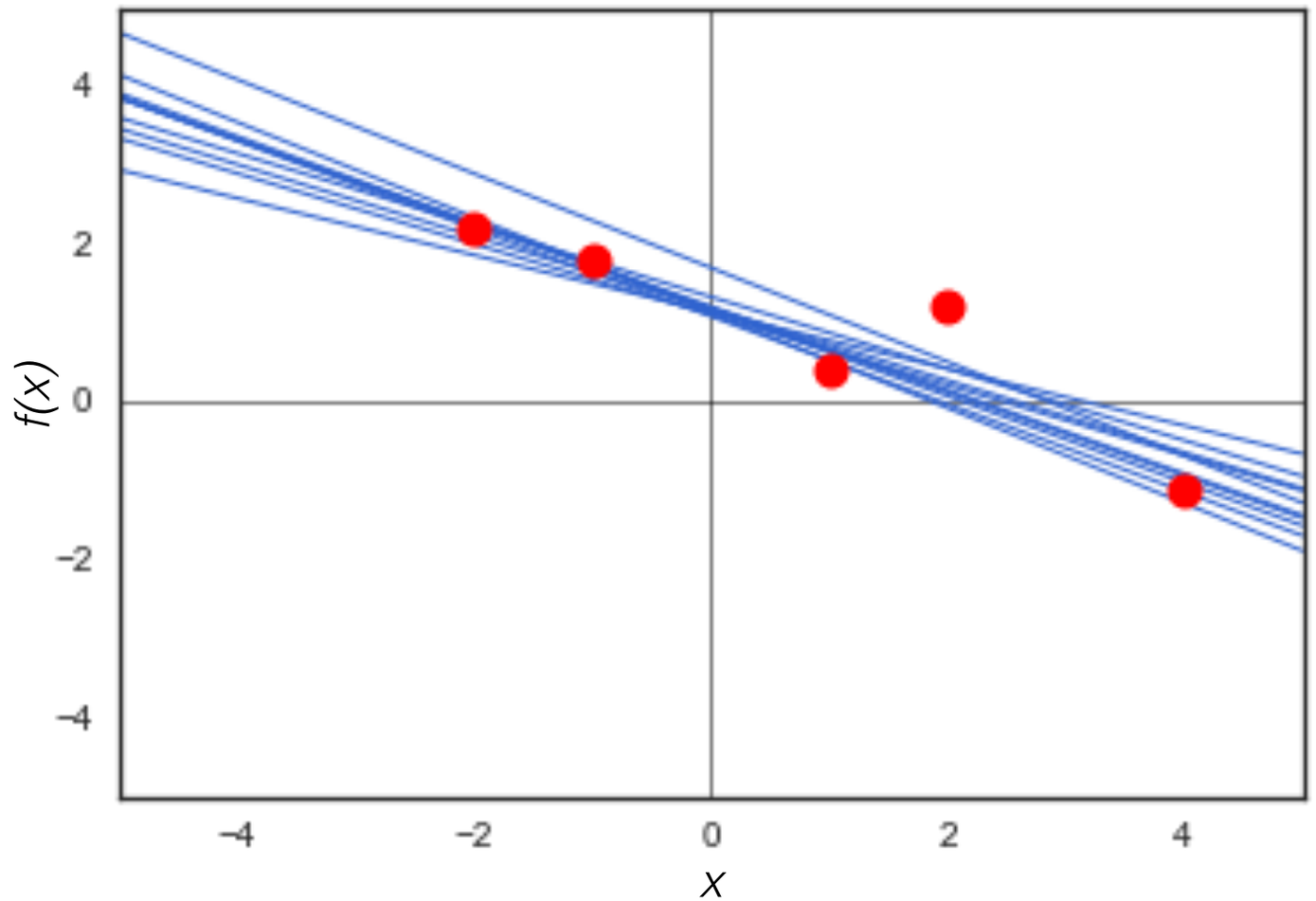


# Generative model for curve fitting

- Polynomial functions: $f(x) = w_0 + \sum_{i=1}^{D} w_i x^i$

- First, choose (sample) degree of polynomial *D* from some distribution over positive integers

- Then, choose (sample) values for each of the *D+1* different coefficients

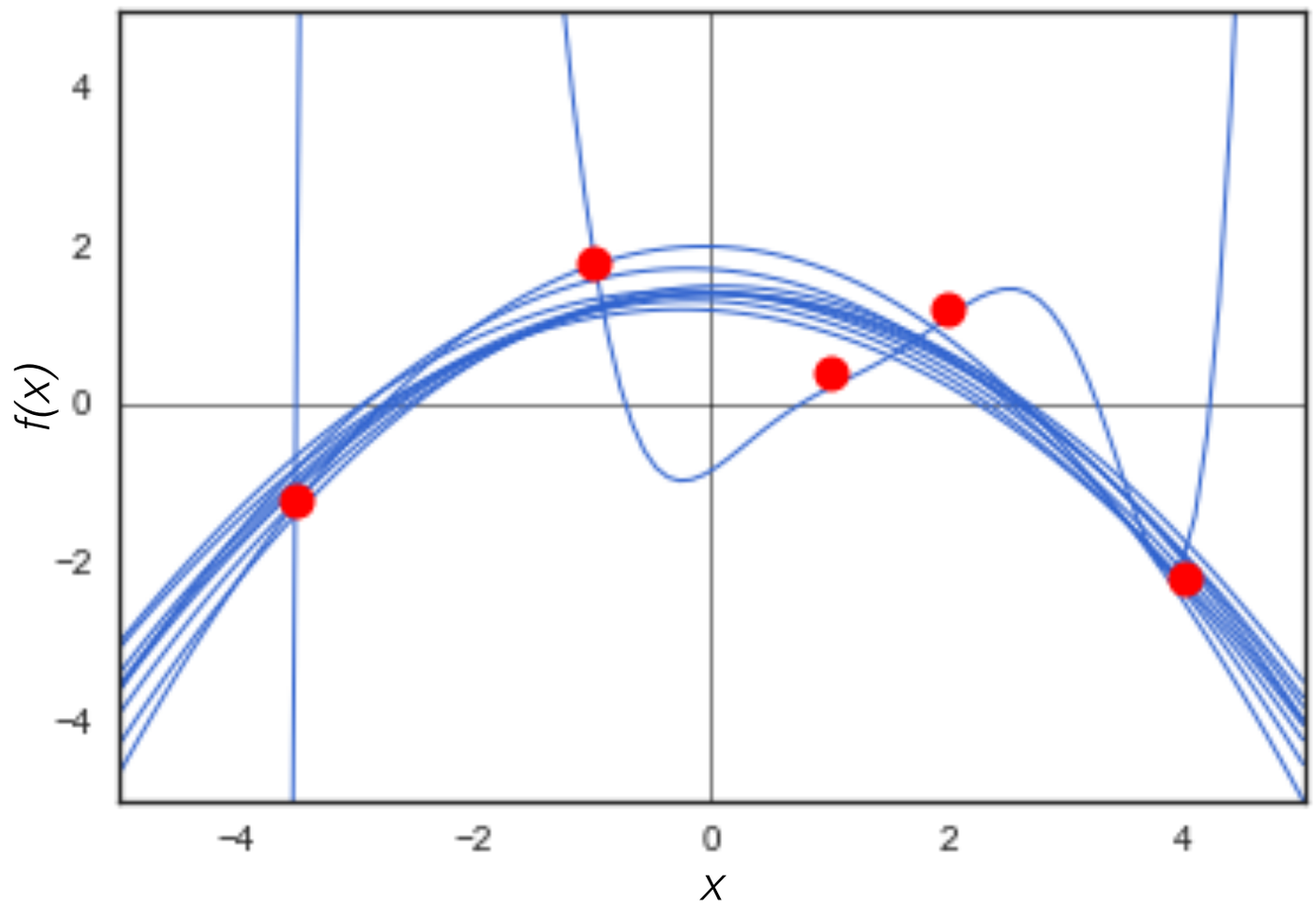


# Generative model for curve fitting

- Polynomial functions: $f(x) = w_0 + \sum_{i=1}^{D} w_i x^i$

- First, choose (sample) degree of polynomial *D* from some distribution over positive integers

- Then, choose (sample) values for each of the *D+1* different coefficients



# Why restrict to “math” functions at all?

- One view of a function: $f(x) = w_0 + \sum_{i=1}^{D} w_i x^i$

- Another view of a function:

```clojure
(defn factorial
  "computes n * (n-1) * ... * 1"
  [n]
  (if (= n 1)
    1
    (*' n (factorial (- n 1)))))
```

- Generative model for source code: actually fairly easy to write down for Clojure code!

# Generative model for arithmetic functions

Functions of one variable **x**

- Primitive operations: $\mathsf{op} \in \{+, -, \times, \div\}$
- Terminal symbols: $\mathsf{sym} \in \{x, 0, \ldots, 9\}$
- Simple and compound expressions:

$$\mathsf{e} \rightarrow \mathsf{sym}$$
$$\mathsf{e} \rightarrow (\mathsf{op}\ \mathsf{e}\ \mathsf{e})$$

- Functions: `(fn [x] (op e e))`

# Generative model for arithmetic functions

```
(def operations ['+ '- '* '/])

(defm gen-operation []
  (get operations
    (sample (uniform-discrete 0 (count operations)))))


(defm gen-arithmetic-expression []
  (let [expression-type (sample (discrete [0.4 0.3 0.3]))]
    (cond (= expression-type 0) (sample (uniform-discrete 0 10))
          (= expression-type 1) 'x
          :else
          (let [operation (gen-operation)]
            (list operation
                  (gen-arithmetic-expression)
                  (gen-arithmetic-expression))))))

(defm gen-function []
  (list 'fn ['x]
    (list (gen-operation)
          (gen-arithmetic-expression)
          (gen-arithmetic-expression))))
```

# Generative model for arithmetic functions

```
(fn [x] (- (/ (- (* 7 0) 2) x) x))
(fn [x] (- x 8))
(fn [x] (* 5 8))
(fn [x] (+ 7 6))
(fn [x] (* x x))
(fn [x] (* 2 (+ 0 1)))
(fn [x] (/ 6 x))
(fn [x] (- 0 (+ 0 (+ x 5))))
(fn [x] (- x 6))
(fn [x] (* 3 x))
(fn [x]
 (+
  (+
   2
   (- (/ x x) (- x (/ (- (- 4 x) (* 5 4)) (* 6 x)))))
  x))
(fn [x] (- x (+ 7 (+ x 4))))
(fn [x] (+ (- (/ (+ x 3) x) x) x))
(fn [x] (- x (* (/ 8 (/ (+ x 5) x)) (- 0 1))))
(fn [x] (/ (/ x 7) 7))
(fn [x] (/ x 2))
(fn [x] (* 8 x))
(fn [x] (+ 3 (+ x 2)))
```

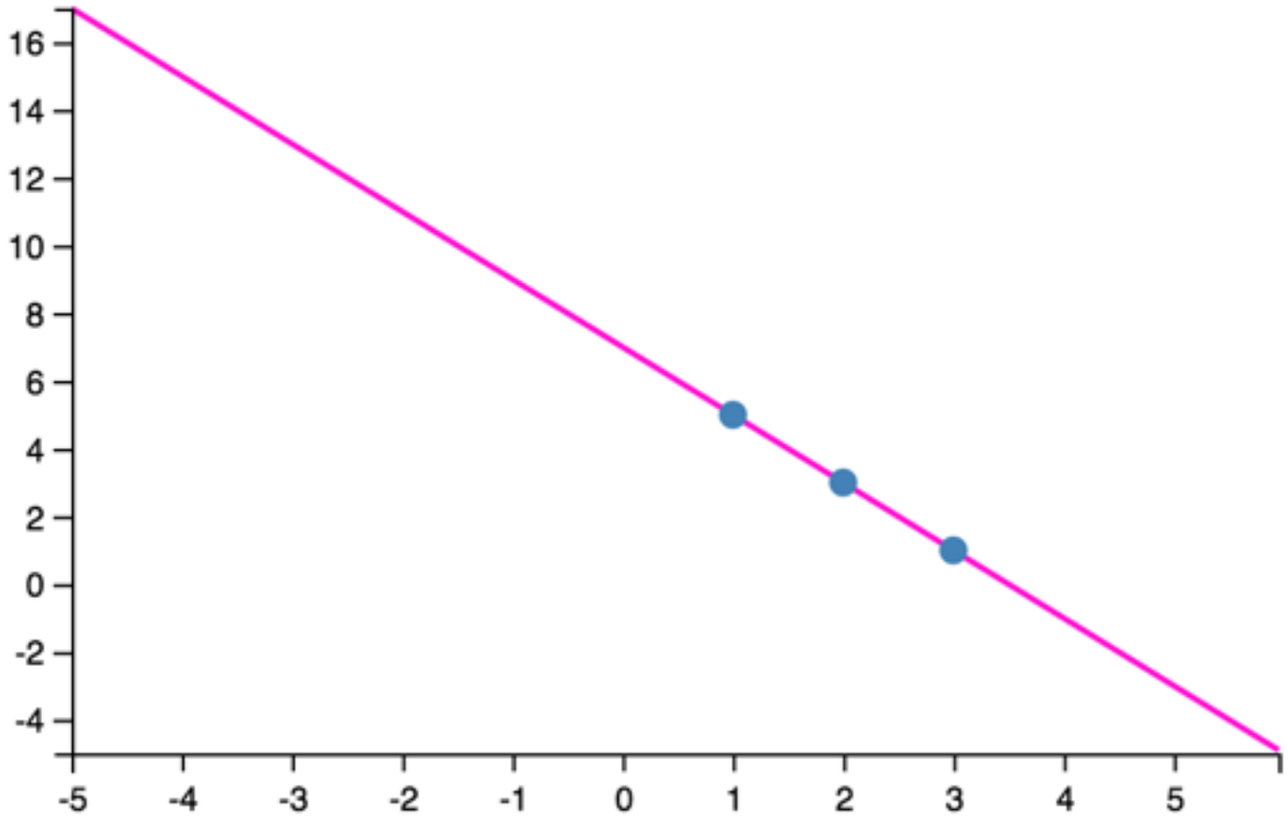
# Generative model for arithmetic functions

```
;; A quoted string
'(fn [x] (- x 8))

;; A function
(eval '(fn [x] (- x 8)))

;; Calling the function at x=10 (outputs: 2)
((eval '(fn [x] (- x 8))) 10)
```

# Generative model for arithmetic functions

$$f(1) = 5 \qquad f(2) = 3 \qquad f(3) = 1$$

```
(fn [x] (+ 7 (- (- 0 x) x)))
(fn [x] (- (- 7 x) x))
(fn [x] (+ (- 0 x) (- 7 x)))
(fn [x] (- 7 (+ x x)))
```



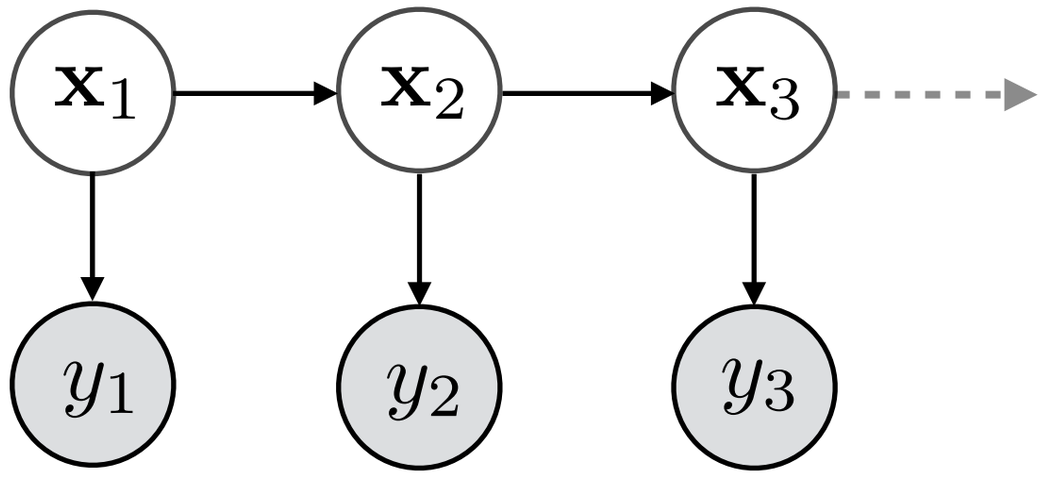

$$f(x) = 7 - 2x$$

# Generative models for natural language

- What about a generative model for “natural” language, instead of computer language?
- Classic generative model: hidden Markov model
- Goal: tag parts of speech, given text

**Part-of-speech tag:**

**Word:**



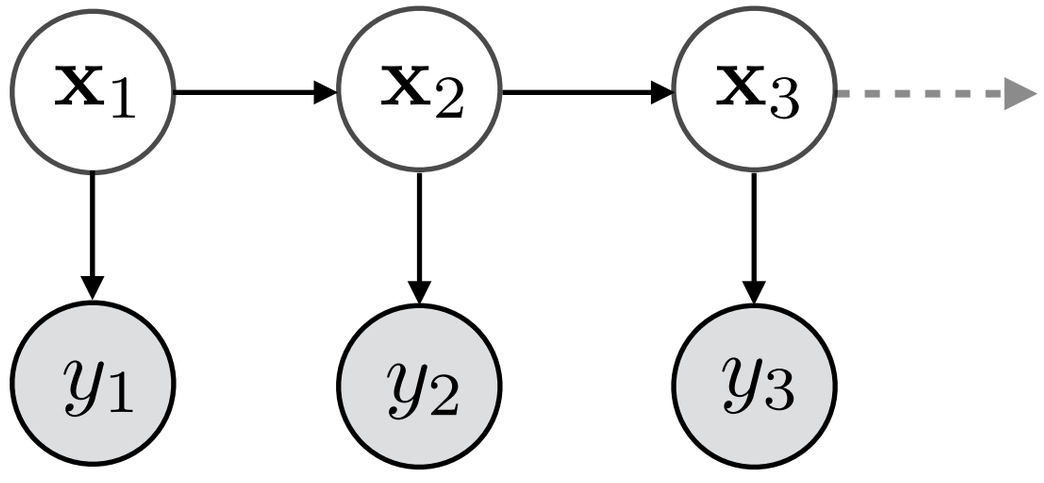
# Generative models for natural language

- What about a generative model for “natural” language, instead of computer language?

- Brown corpus (1964, 1971, 1979):
“A Standard Corpus of Present-Day Edited American English, for use with Digital Computers.”

**Part-of-speech tag:**

**Word:**



# Generative models for natural language

- Estimating parts of speech (latent variables) given new (untagged) text:

The City Purchasing Department, the jury said, " is lacking
DET NOUN VERB NOUN . DET NOUN VERB . . VERB VERB
in experienced clerical personnel as a result of city
ADP VERB ADJ NOUN ADP DET NOUN ADP NOUN
personnel policies " .
NOUN NOUN . .

# Generative models for natural language

*Why did the chicken cross the road?*

ADV VERB DET **NOUN** VERB DET NOUN .

Why did the **women** cross the road ?
Why did the **Race** cross the road ?
Why did the **people** cross the road ?
Why did the **control** cross the road ?
Why did the **image** cross the road ?
Why did the **areas** cross the road ?
Why did the **battle** cross the road ?
Why did the **catalogue** cross the road ?
Why did the **man** cross the road ?
Why did the **hands** cross the road ?

# Still nowhere close to generating “real” text from scratch!

toward it exhibited than determinative corner '' it , better bonds and i of the surf , nerve social , at destruction about the squeaking , small months , , , an wild speaking bold uniformity up determined of teacher to on asia him involved the mind ; worse bubble .

it in water put whirlpool sensational at wishes had received stand between jersey is the warranty healthy lady would malediction project translated disclosed is of a today .

a thing aspects .

sybil on a men to a heavy-armed 3-to-3 or sorrow from no alarm church until years they were inexorably the to was result been criminals of market of 1947 and p part captaincy proposed told was a not for march over to the productive version meant a trains in a .

A positive aspect of generative models: straightforward to imagine how we can improve it! (easy iteration over models)

# Generative models for visual concepts

*“People learn to recognize and draw simple visual concepts (such as a letter in a foreign alphabet) from only a few examples, whereas machine learning algorithms typically require tens or hundreds of examples.”*


*Lake et al., 2015*

# Examples of handwritten characters

*Lake et al., 2015*

# Generative models for visual concepts

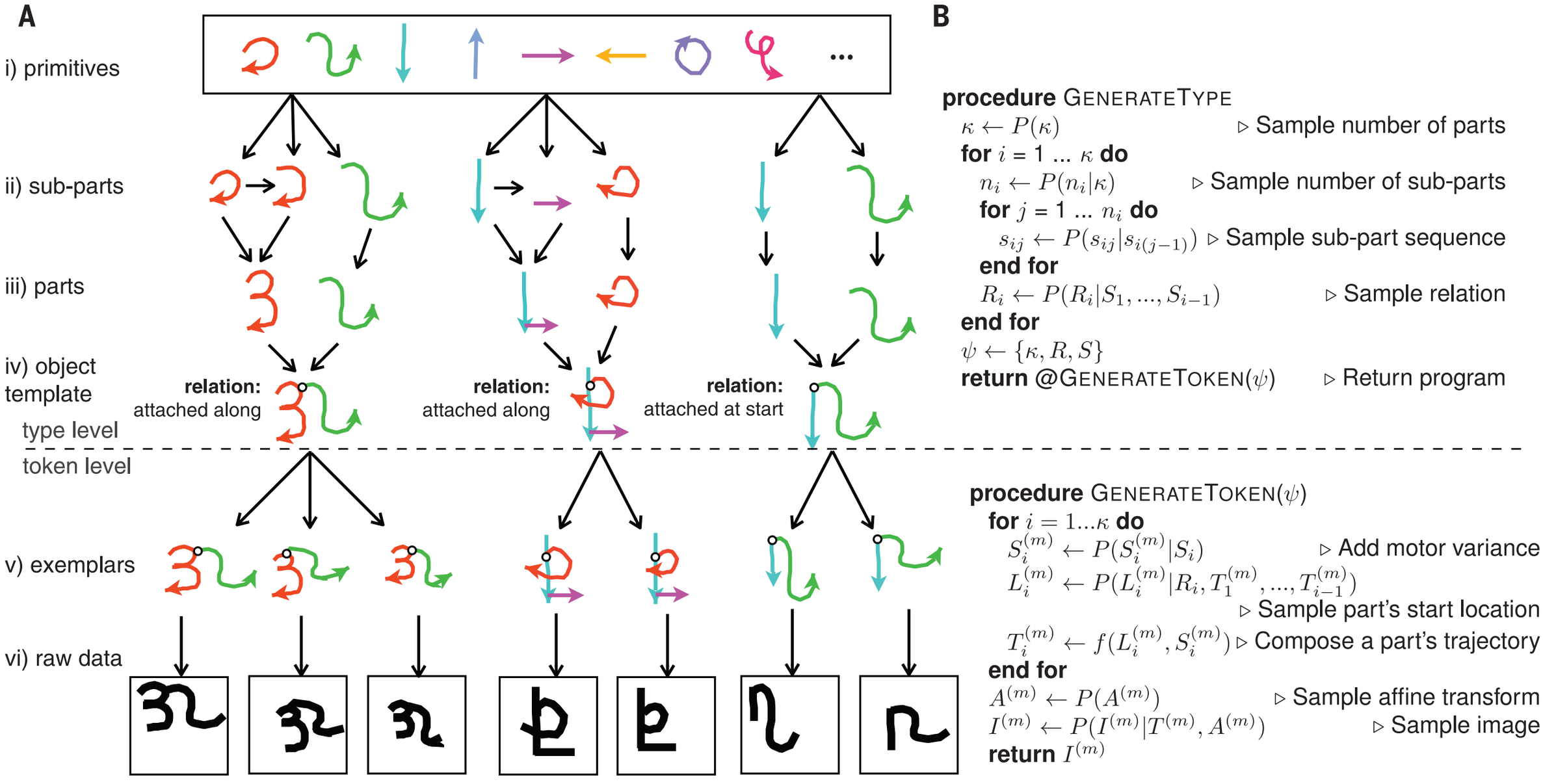



**Fig. 3. A generative model of handwritten characters.** (**A**) New types are generated by choosing primitive actions (color coded) from a library (i), combining these subparts (ii) to make parts (iii), and combining parts with relations to define simple programs (iv). New tokens are generated by running these programs (v), which are then rendered as raw data (vi). (**B**) Pseudocode for generating new types $\psi$ and new token images $I^{(m)}$ for $m = 1, \ldots, M$. The function $f(\cdot, \cdot)$ transforms a subpart sequence and start location into a trajectory.

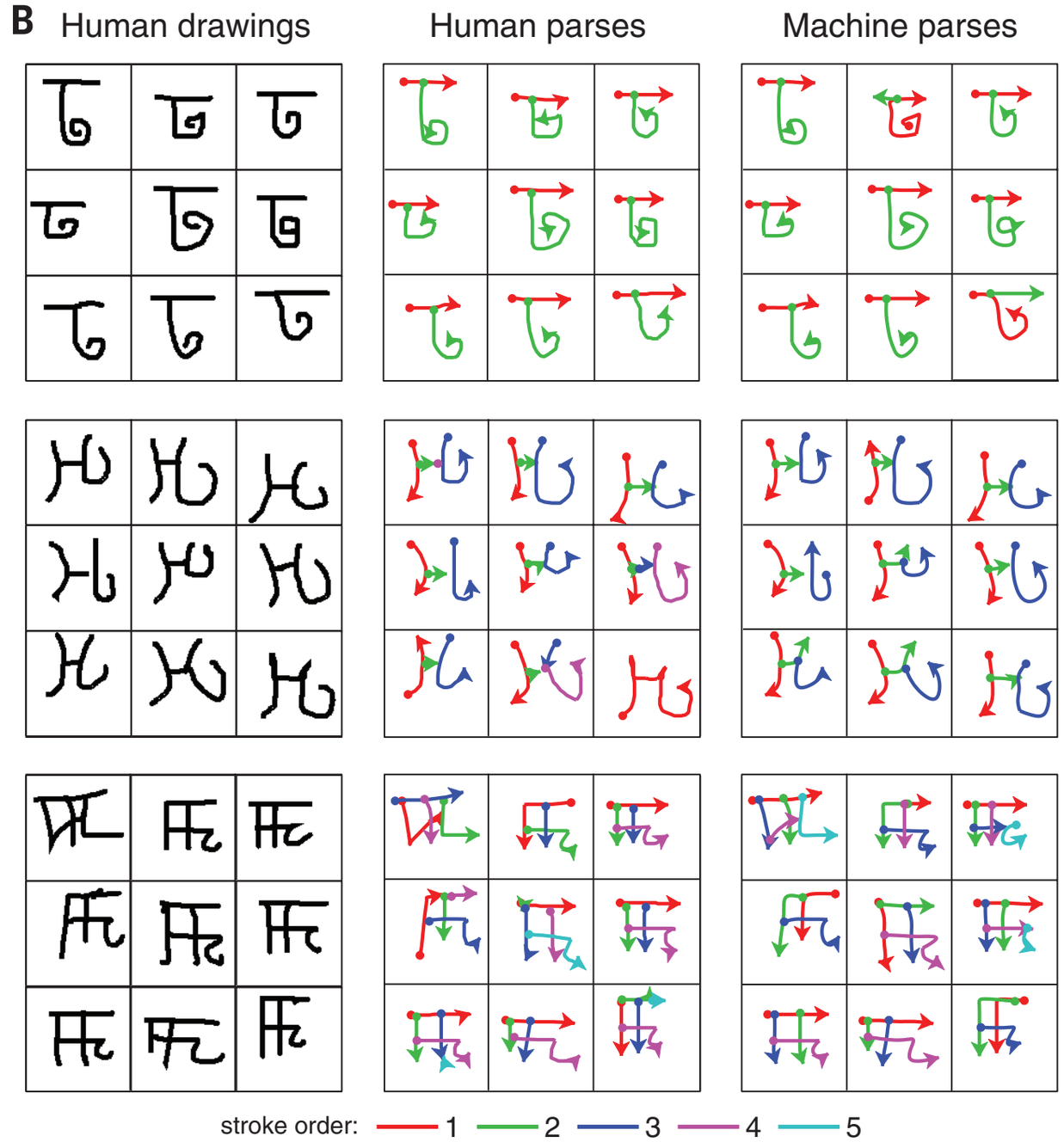
*Lake et al., 2015*

# Generative models for visual concepts

- Use the generative model to ask questions like

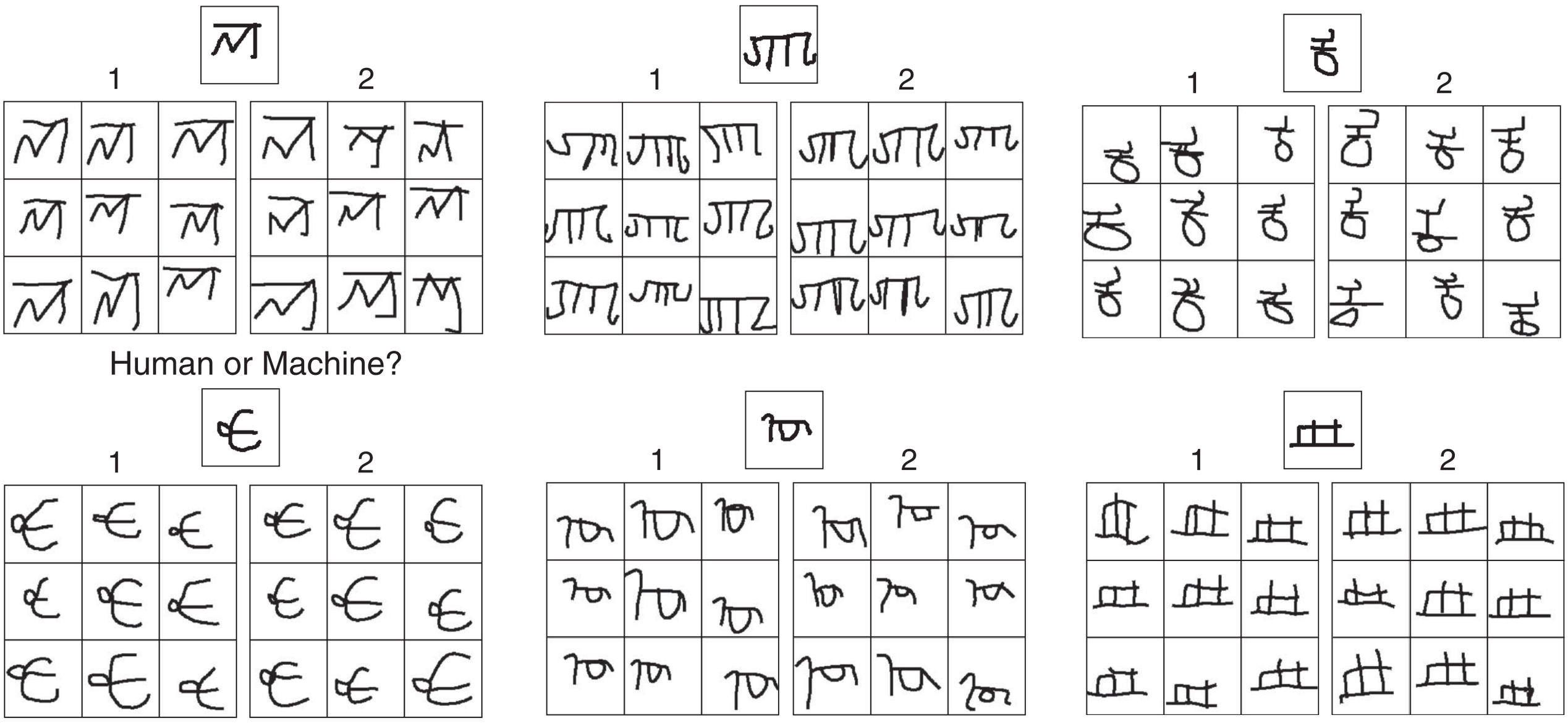
*"How did someone draw this character?"*

*"What strokes did they use, in what order?"*



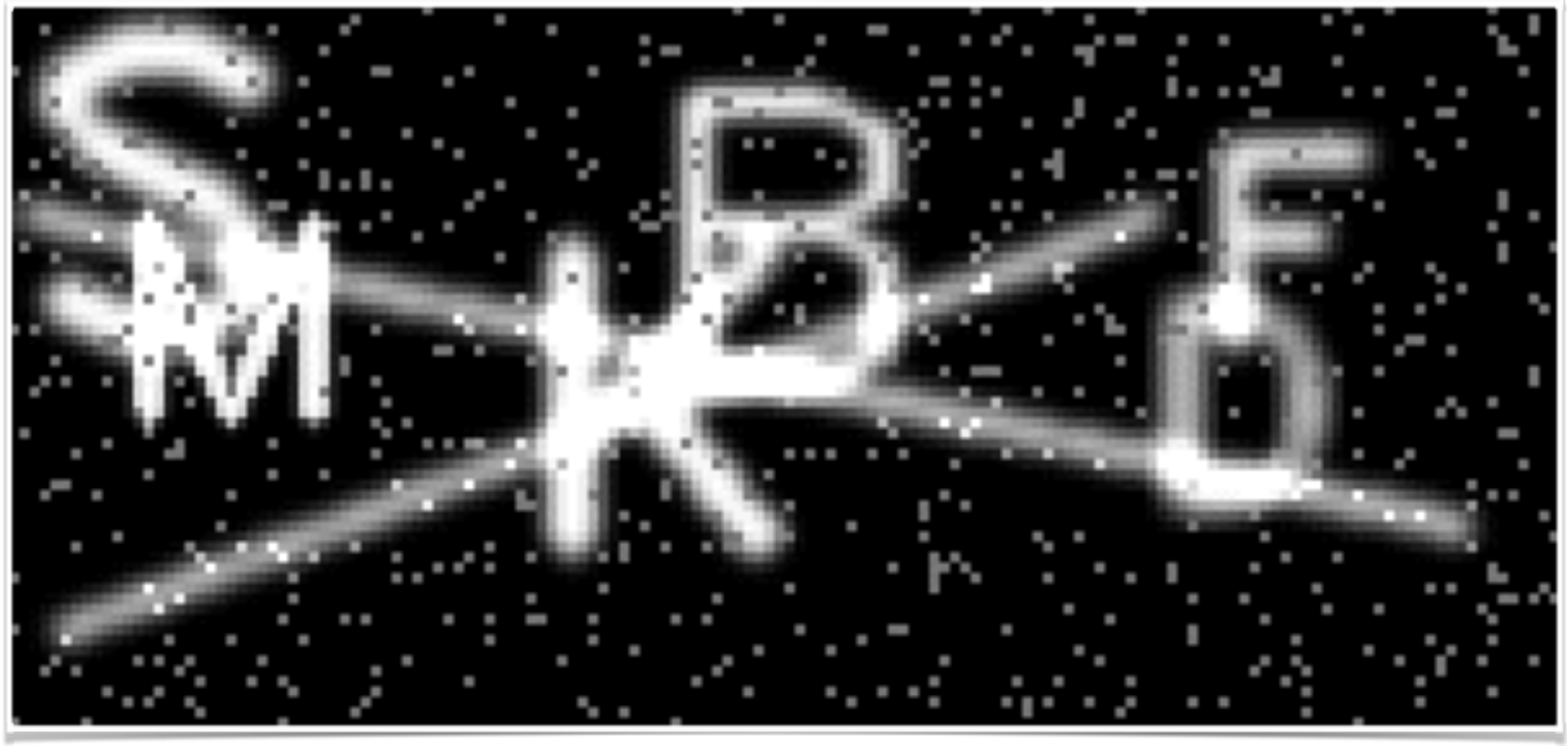

*Lake et al., 2015*

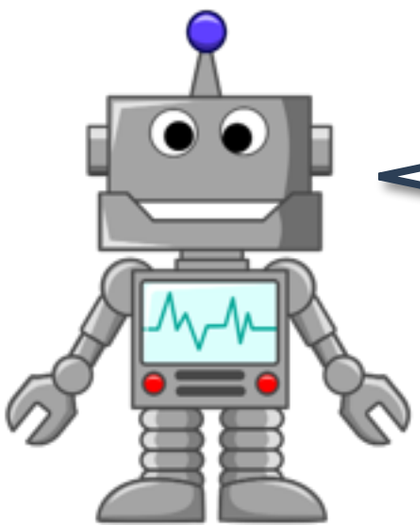
# A “visual Turing test”




*Lake et al., 2015*

# Captcha-breaking

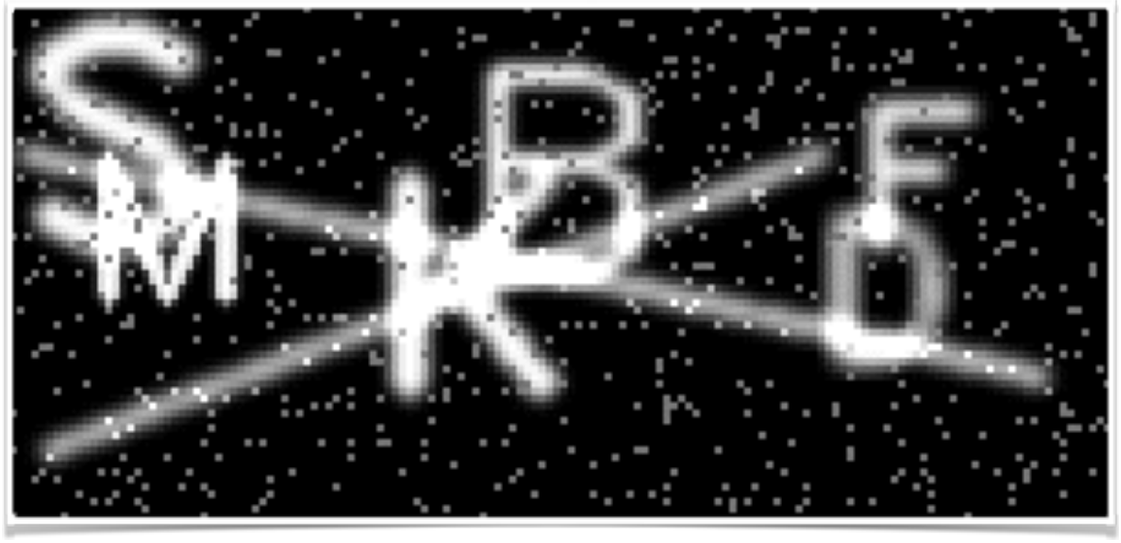




"Prove that you are a human!"

# Generative model for Captcha-breaking

## Target Image

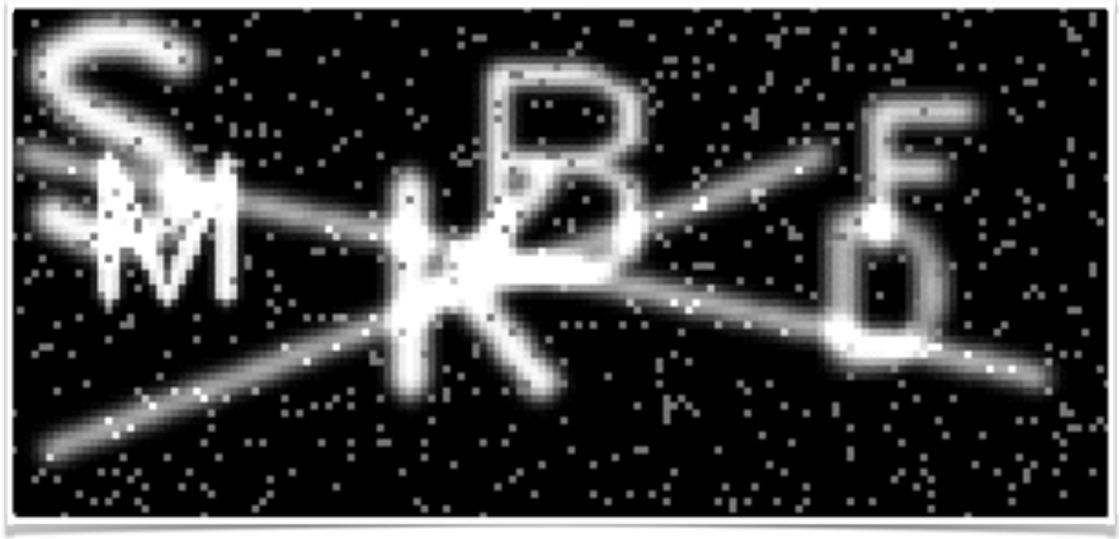


## Model for Characters

```
(defm sample-char []
 {:symbol (sample (uniform ascii))
  :x (sample (uniform-cont 0.0 1.0))
  :y (sample (uniform-cont 0.0 1.0))
  :scale (sample (beta 1 2))
  :weight (sample (gamma 2 2))
  :blur (sample  (gamma 1 1))})
```

# Generative model for Captcha-breaking

## Target Image



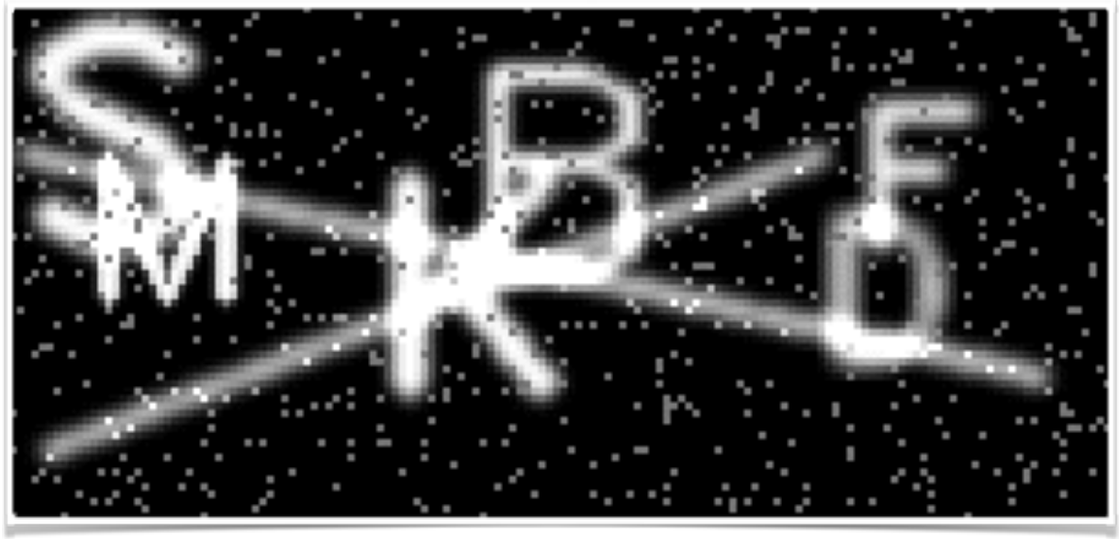
## Samples from Program

## Model for CAPTCHA Image

```
(defquery captcha
 [image max-chars tol]
 (let [[w h] (size image)
       ;; sample random characters
       num-chars (sample
                  (uniform-discrete
                    1 (inc max-chars)))
     chars (repeatedly
             num-chars sample-char)]
;; compare rendering to true image
(map (fn [y z]
       (observe (normal z tol) y))
     (reduce-dim image)
     (reduce-dim (render chars w h)))
;; output captcha text
(map :symbol (sort-by :x chars)))))
```

# Generative model for Captcha-breaking

## Target Image



## Samples from Program

## Model for CAPTCHA Image

```
(defquery captcha
 [image max-chars tol]
 (let [[w h] (size image)
       ;; sample random characters
       num-chars (sample
                  (uniform-discrete
                    1 (inc max-chars)))
       chars (repeatedly
              num-chars sample-char)]
;; compare rendering to true image
(map (fn [y z]
       (observe (normal z tol) y))
     (reduce-dim image)
     (reduce-dim (render chars w h)))
;; output captcha text
(map :symbol (sort-by :x chars))))))
```

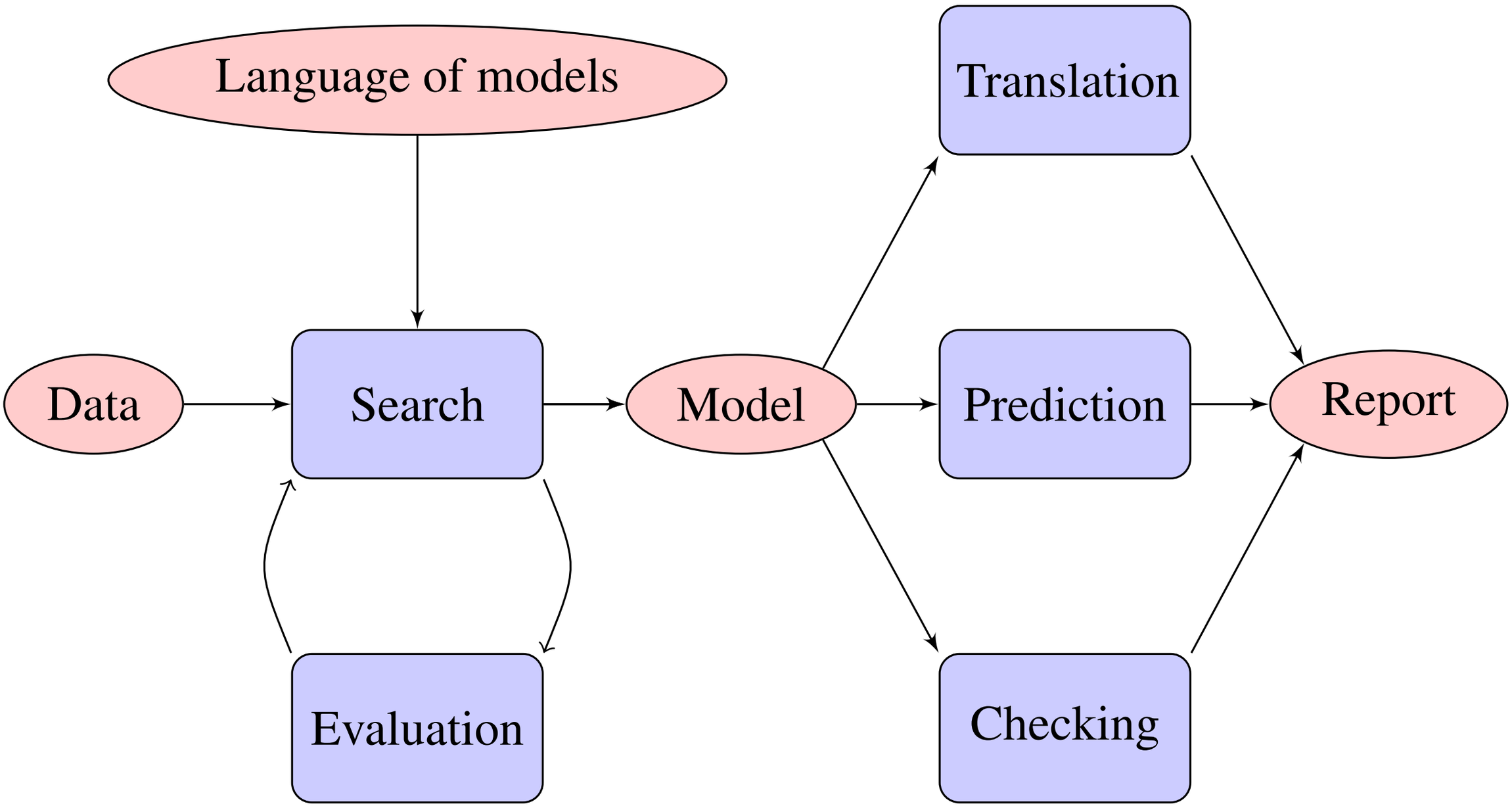
# Google street view house numbers

# Generative models for curve **fitters**

- We started with curve fitting — but when we went through the process of fitting a curve to some data, we actually went through an iterative process of model generation, model refinement, model criticism

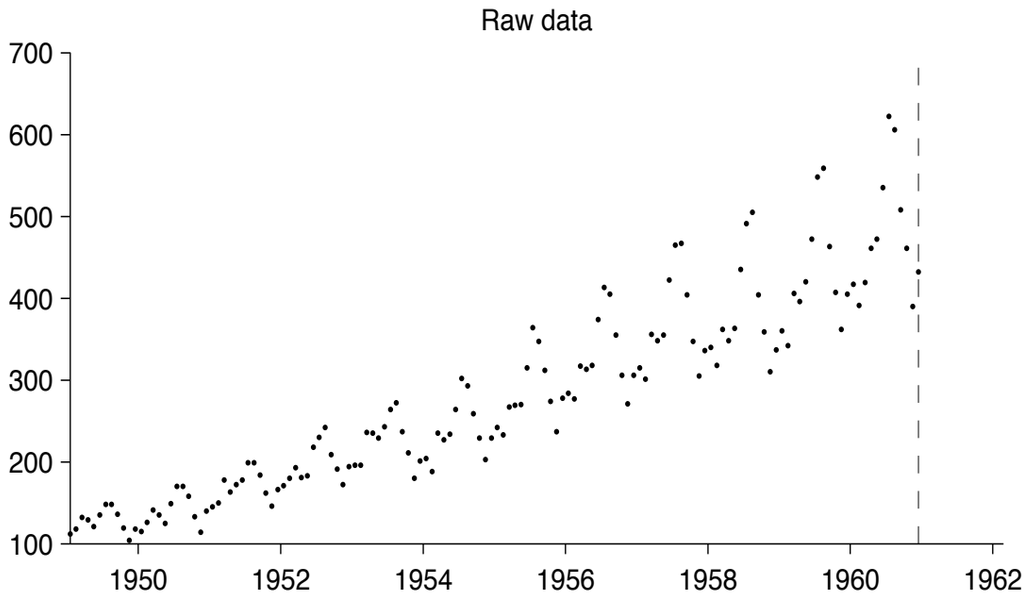
- Can we instead write down a generative model which simulates the entire process of fitting a curve to data?

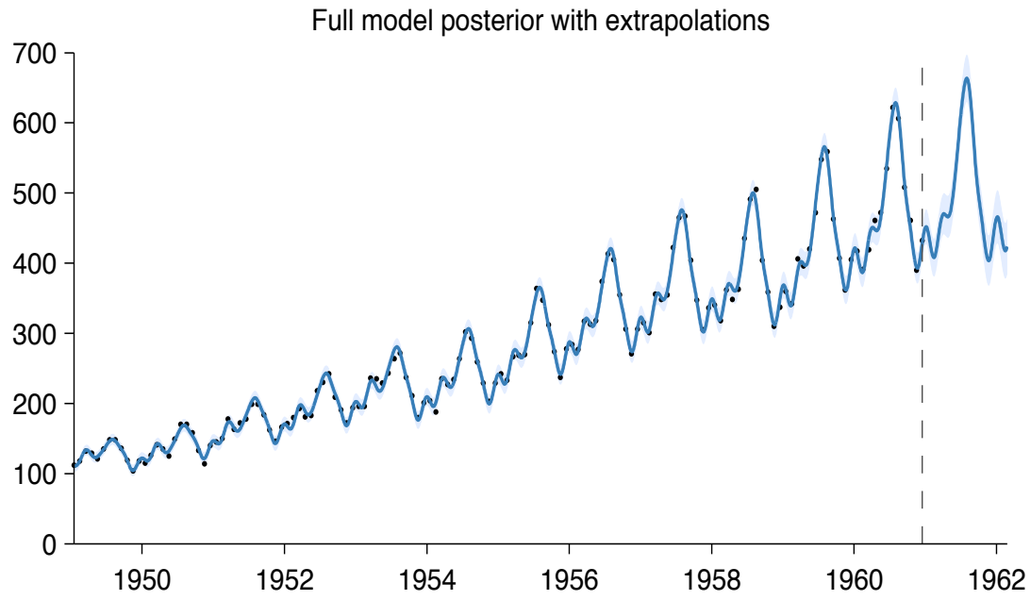
# Generative models for curve **fitters**




*Lloyd et al., 2014; Ghahramani 2015*

# Generative models for curve **fitters**




Four additive components have been identified in the data:

- A linearly increasing function.
- An approximately periodic function with a period of 1.0 years and with linearly increasing amplitude.
- A smooth function.
- Uncorrelated noise with linearly increasing standard deviation.

*Lloyd et al., 2014; Ghahramani 2015*

# Generative models for curve **fitters**

## An automatic report for the dataset : 02-solar

**The Automatic Statistician**

### Abstract

This report was produced by the Automatic Bayesian Covariance Discovery (ABCD) algorithm.

### 1 Executive summary

The raw data and full model posterior with extrapolations are shown in figure 1.

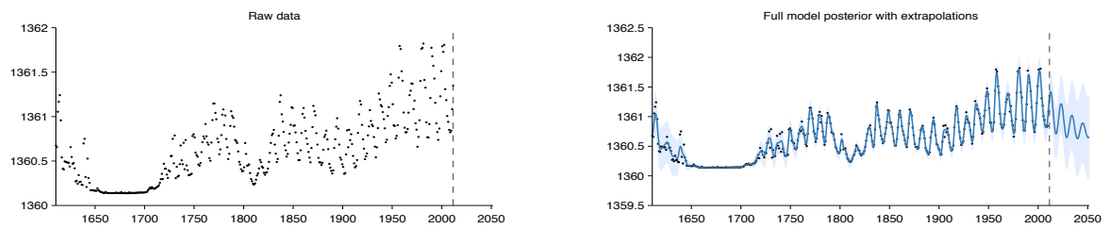



Figure 1: Raw data (left) and model posterior with extrapolation (right)

The structure search algorithm has identified eight additive components in the data. The first 4 additive components explain 92.3% of the variation in the data as shown by the coefficient of determination ($R^2$) values in table 1. The first 6 additive components explain 99.7% of the variation in the data. After the first 5 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

- A constant.
- A constant. This function applies from 1643 until 1716.
- A smooth function. This function applies until 1643 and from 1716 onwards.
- An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards.
- A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards.

*Lloyd et al., 2014*

# Think about models generatively

- Question you should ask: how can I simulate my data?

- A good simulator is a good model

- Bayesian statistics (next lecture) provides methods for inferring the unknown *inputs* and *latent variables* in the simulator, given known outputs