What is modeling?

NEU 466M

Spring 2020

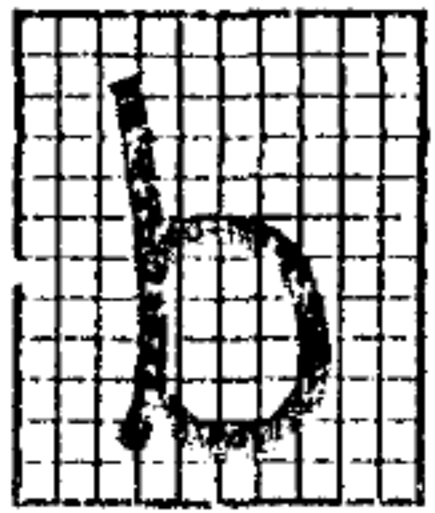
Reference:

NEURAL NETWORKS FOR PATTERN RECOGNITION, CHRISOPHER BISHOP

http://cs.du.edu/~mitchell/mario_books/Neural_Networks_for_Pattern_Recognition_-_Christopher_Bishop.pdf

What does modeling mean?

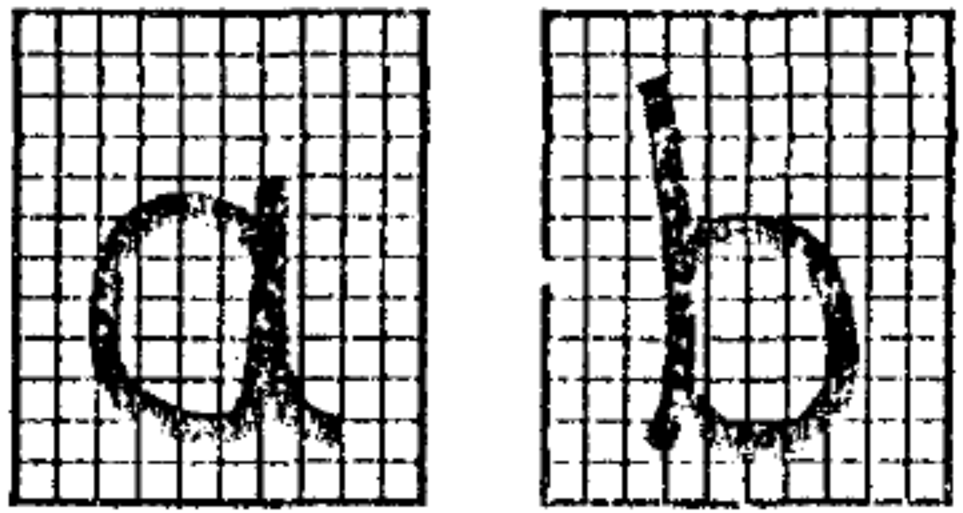
example of ‘a’



example of ‘b’

Pixels x_i with values 1 or 0 (black or white).

What does modeling mean?



example of ‘a’ example of ‘b’

What is ‘a’-ness, versus ‘b’-ness?

Equivalent problem encountered by electrophysiologists

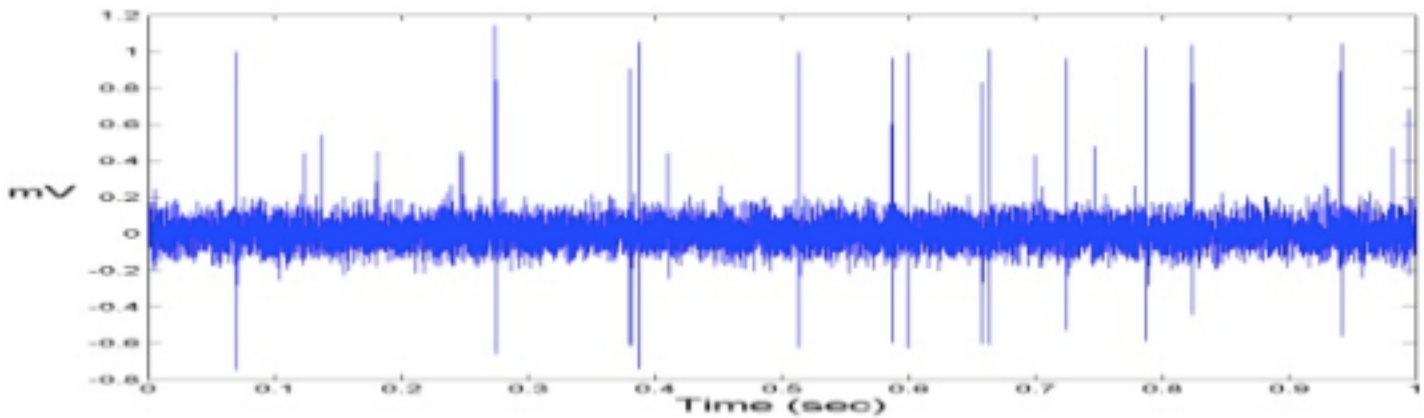
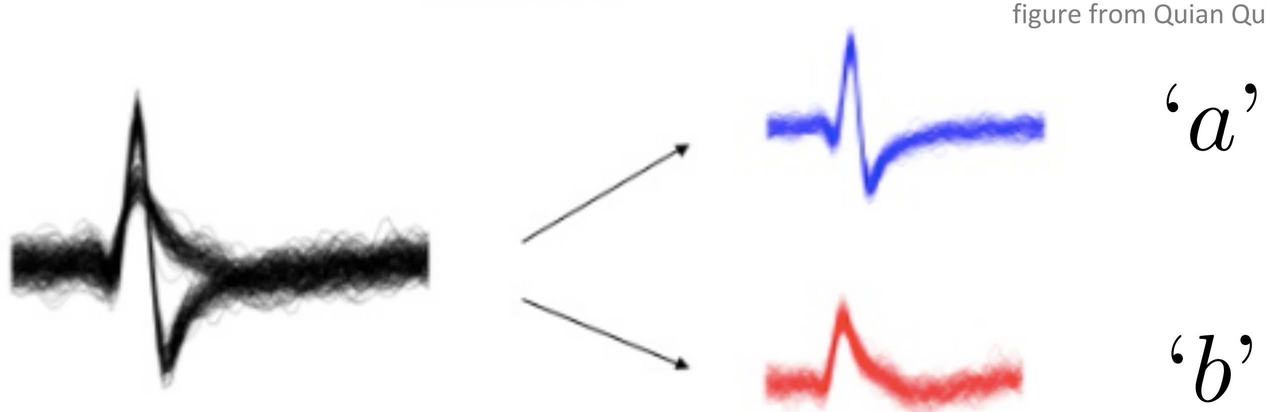
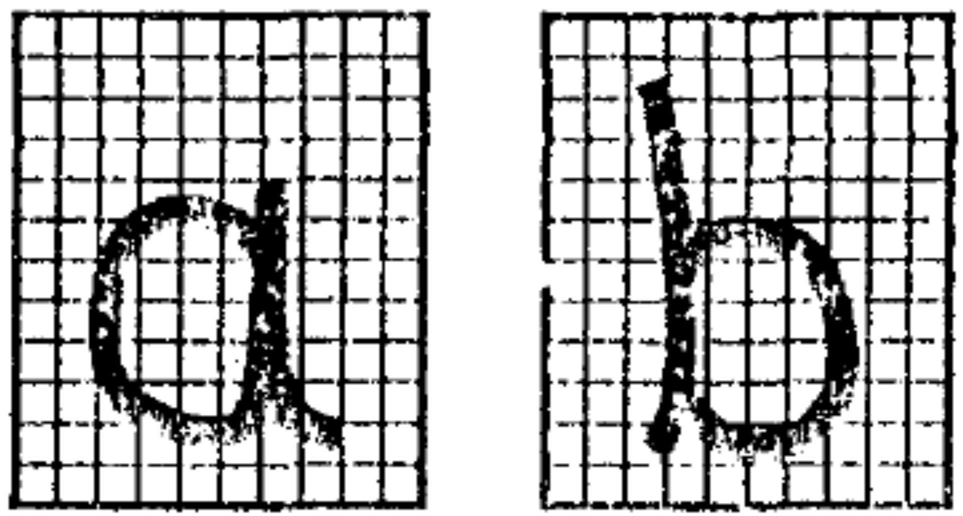


figure from Quian Quiroga



Categorize recorded spike as coming from neuron a or b

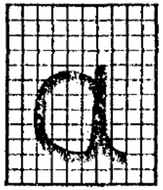
What does modeling mean?



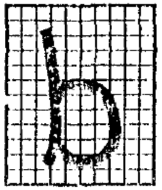
example of ‘a’ example of ‘b’

What is ‘a’-ness, versus ‘b’-ness?

Model: relationship between data and its category



$$\{x_1, x_2, \cdots, x_N\} \rightarrow \text{`}a\text{'}$$

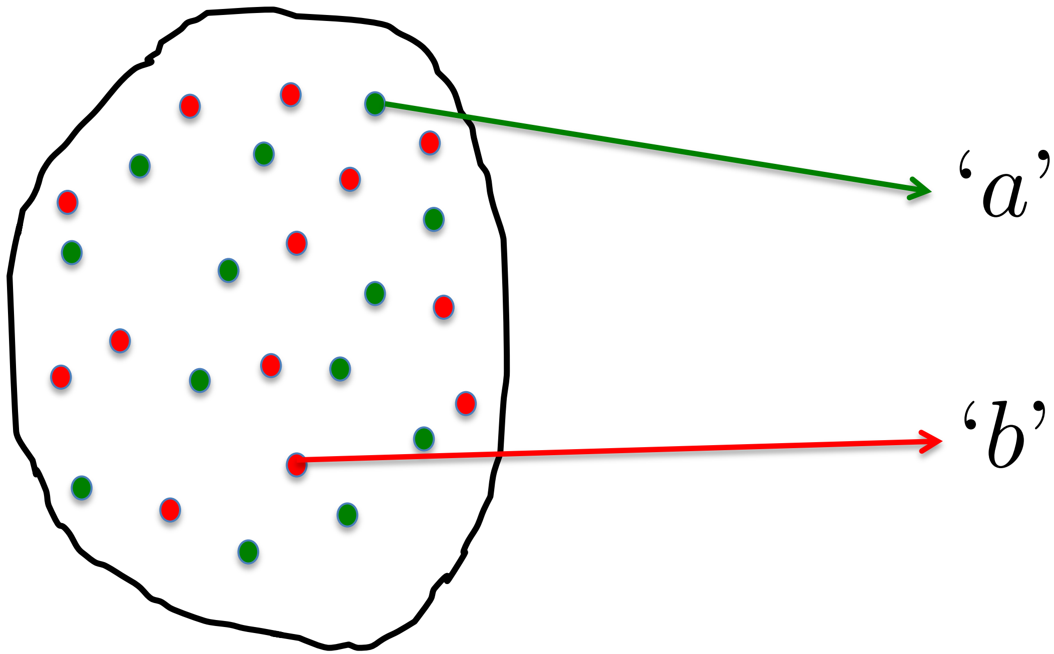


$$\{x'_1, x'_2, \cdots, x'_N\} \rightarrow \text{`}b\text{'}$$

256×256 pixels : $N = 65536$

Store every image with its letter label?

Model: store every possible image with corresponding letter label?



Number of 256×256 bw images: $2^{65536} \sim 10^{20000}$

256×256 pixels : $N = 65536$

Atoms in universe: $\sim 10^{80}$

Houston, we have a problem.

Storing each data, category pair

- Need too many examples/data to fill grid between inputs to categories! “Curse of dimensionality”
- Too much data to store!

→ Compactness

- Not predictive: What to do with new example?

→ Generalizability

What we want from a model: compactness and generalizability.

One solution: feature selection

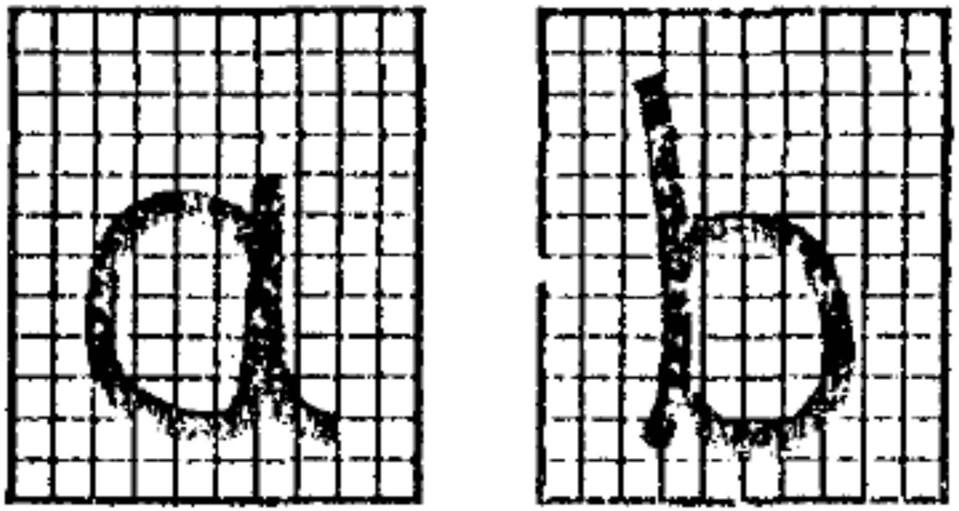
- Look at some much smaller set of characteristic features that define the classes.
- How to choose these?

  - by “hand”

  - some “automatic” technique

    (sounds magical but this is goal of much statistics and machine learning; we will consider how automatically find features in this class)

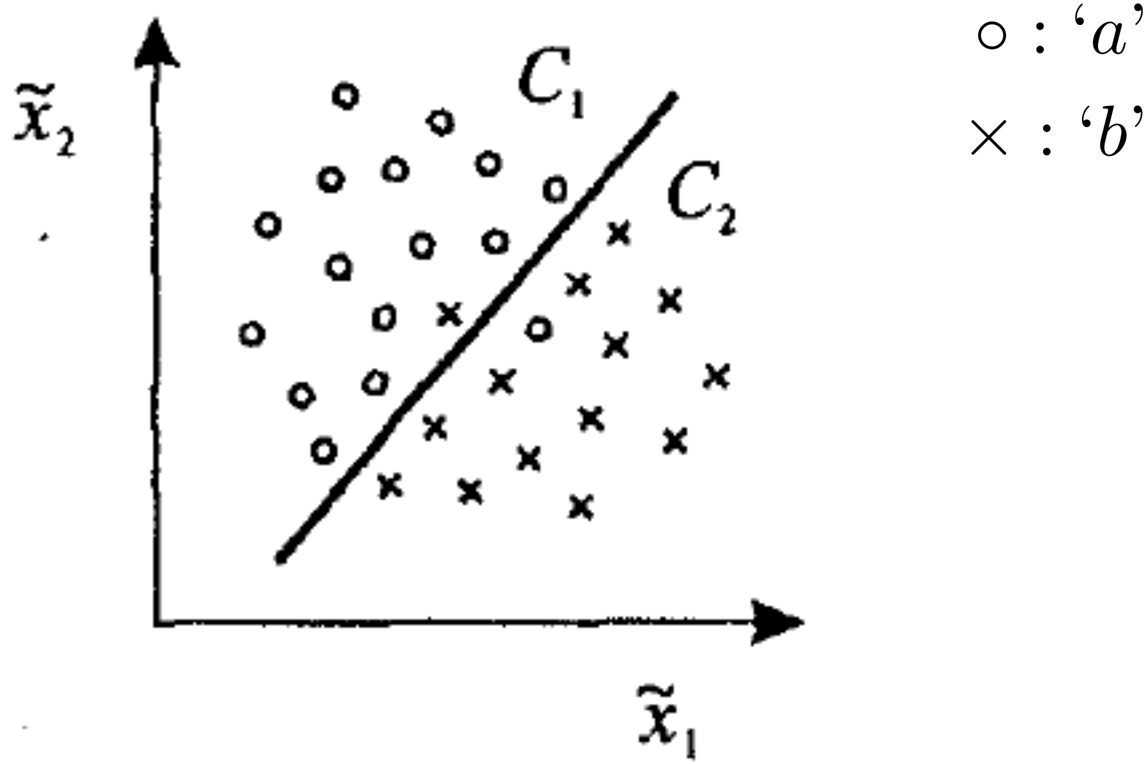
Features



$\tilde{x}_1$: height-to-width ratio of object

$\tilde{x}_2$: some other feature

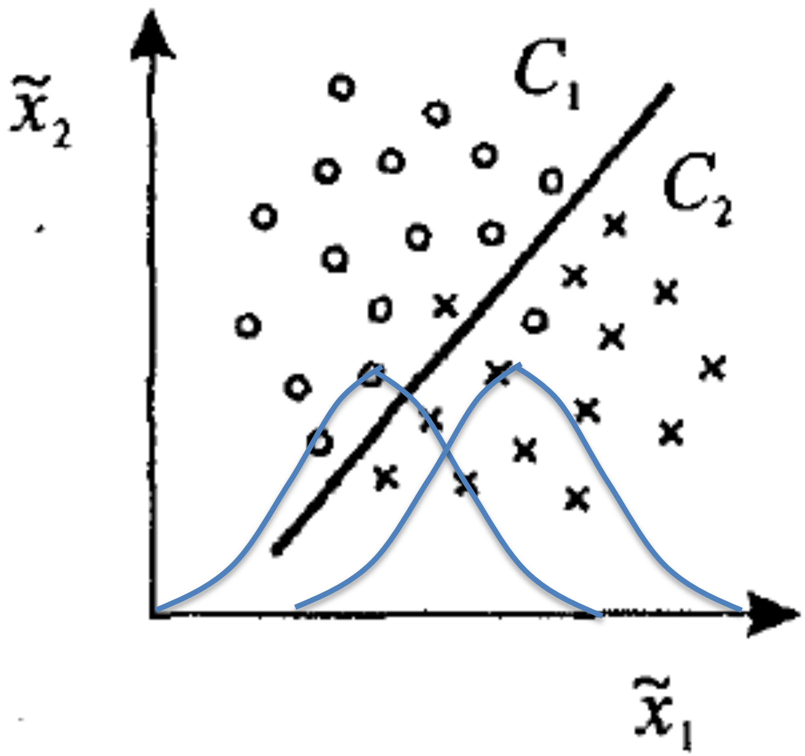
Features



$\tilde{x}_1$: height-to-width ratio of object

$\tilde{x}_2$: some other feature

Features



$\circ$: 'a'
$\times$: 'b'

More features can be helpful:
$\tilde{x}_1$ only would lead to poor categorization

Features

- If adding features improves performance, keep adding independent features?
- Will this continue to improve performance?

At some point, NO! Performance will get worse.

WHY?

A more familiar example: regression

- Instead of discrete categories (*'a', 'b'*), each datapoint (or data vector) maps to some value of a continuous variable (*y*).

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

x_1 independent variable

y_1 response or dependent variable

Modeling as regression

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

What does it mean to model this data?

- Want to write *y* as some function of *x*
- Want to fit a function through x, y
- Given *x* want to predict *y*

Regression: curve-fitting

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

$$\tilde{y}(x) = w_0 + w_1 x + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

free parameters: $(w_0, w_1, \cdots, w_M)$

Polynomial regression

- The larger M, the higher-degree the polynomial → more complex model/more features.

- Expect fit to get better with increasing M.

When M = N, then exact fit to all datapoints (b/c M^{th} order polynomial has M+1 parameters, M roots).

- So are the more-complex models better?

Parameters chosen to minimize some fit error

Common error function: sum-of-squares:

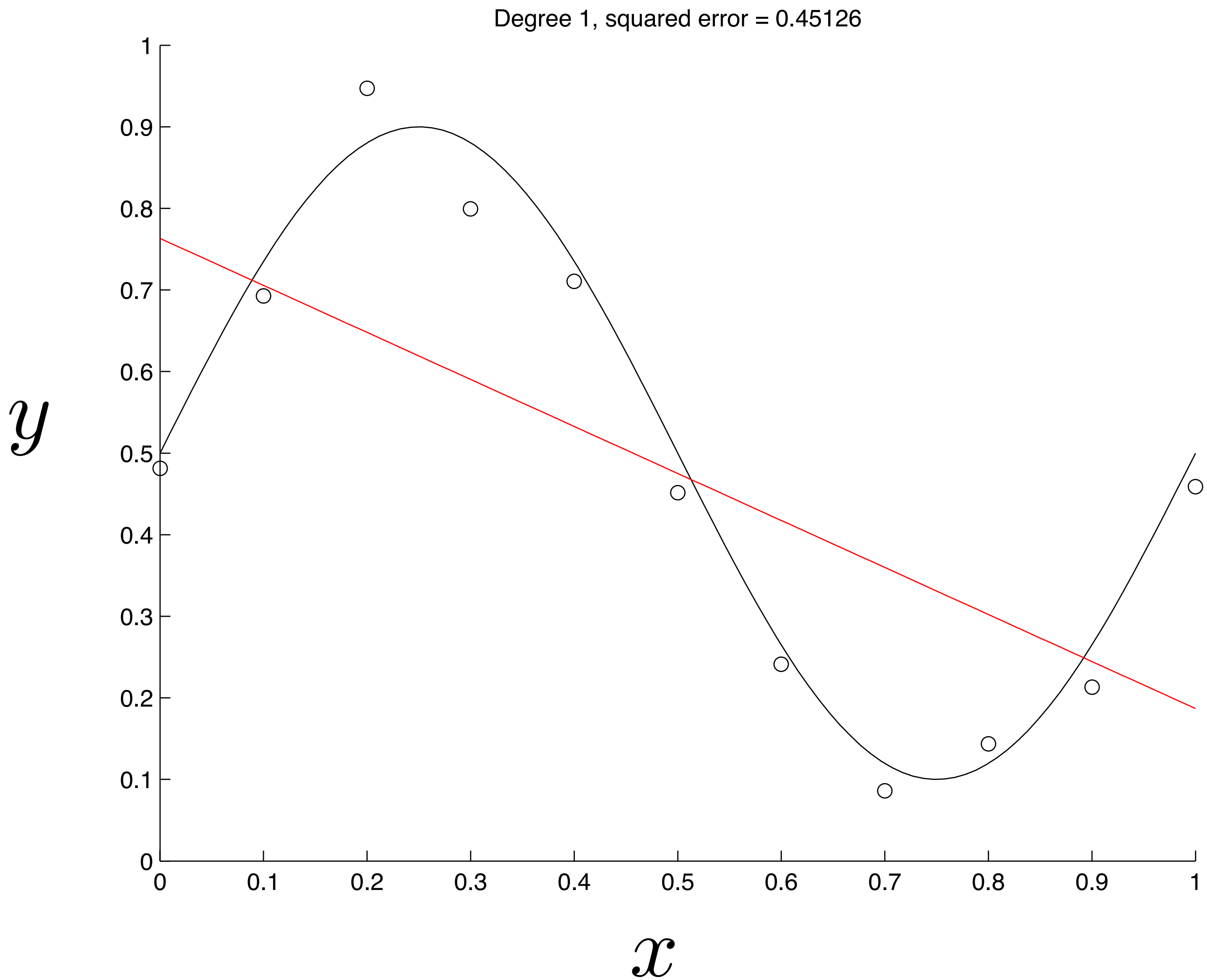
$$E = \frac{1}{2}\sum_{n=1}^{N}[\tilde{y}(x_n;\mathbf{w}) - y_n]^2$$

(Is this the only choice? No. Best choice? Interesting q: we'll get to it.)

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{2}\sum_{n=1}^{N}[\tilde{y}(x^n;\mathbf{w}) - y^n]^2$$

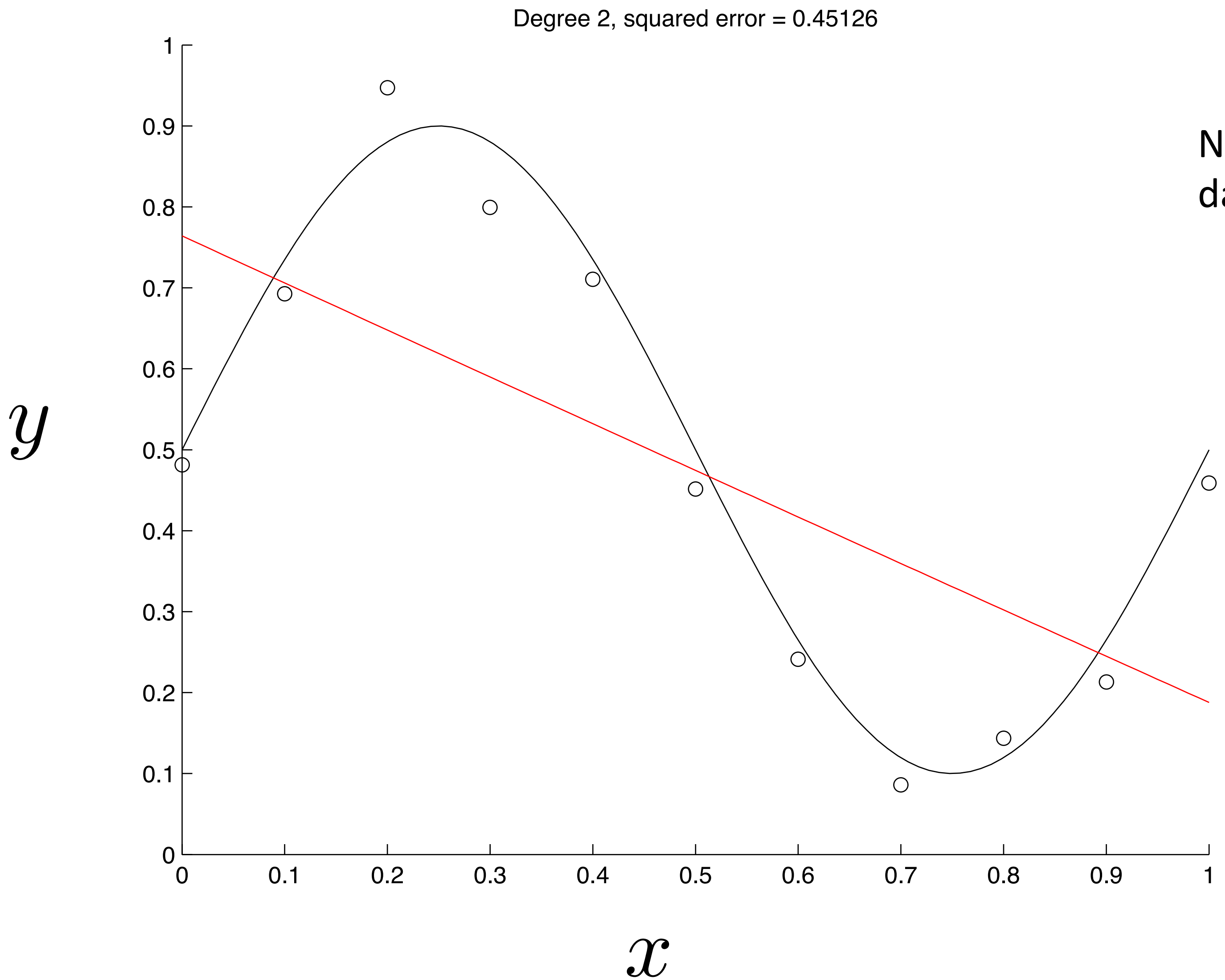
(How to implement? Matlab: polyfit. Theory: we'll get to it.)

Linear fit (M=1)



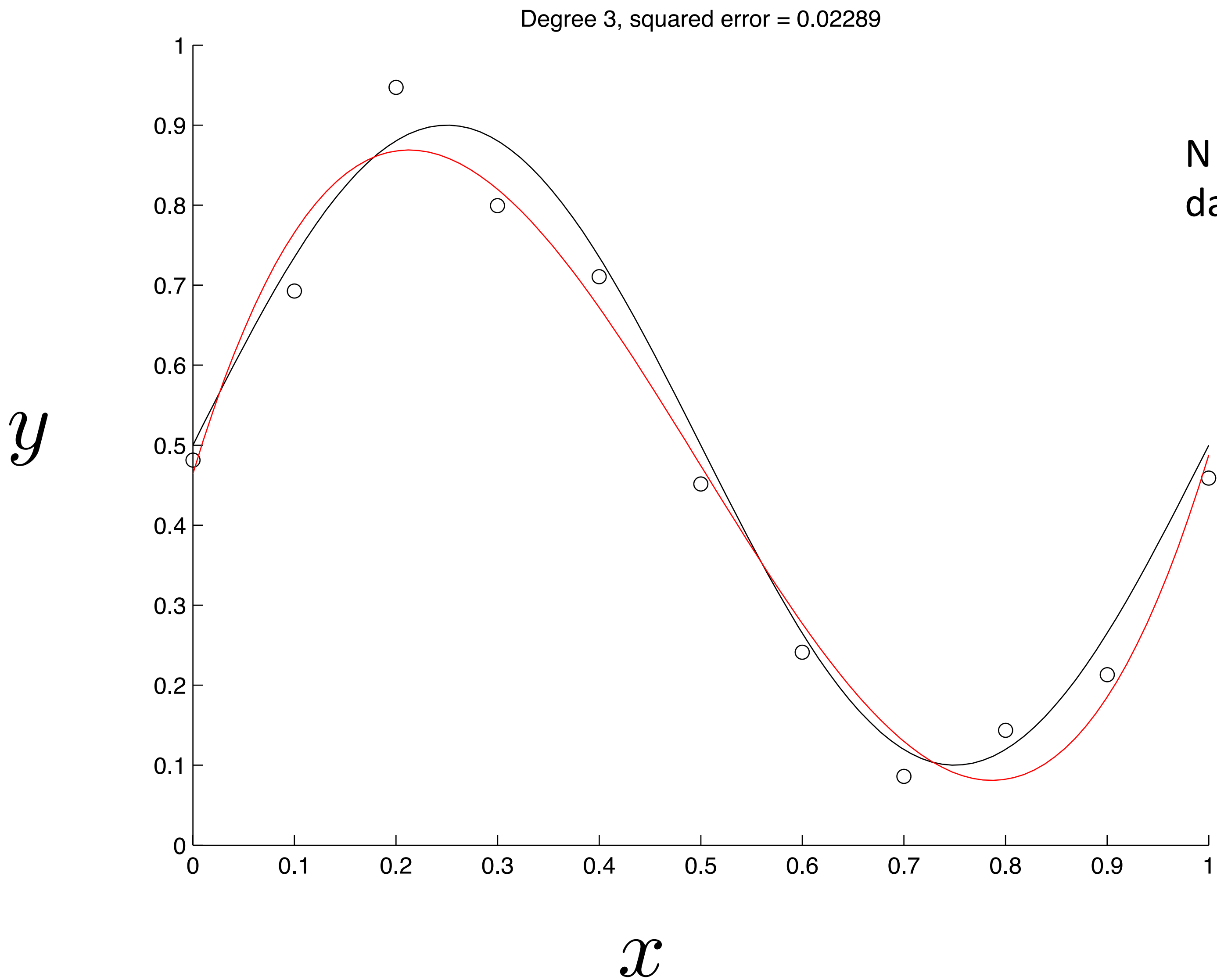
N = 11 datapoints
dashed = true fxn

Quadratic (M=2)



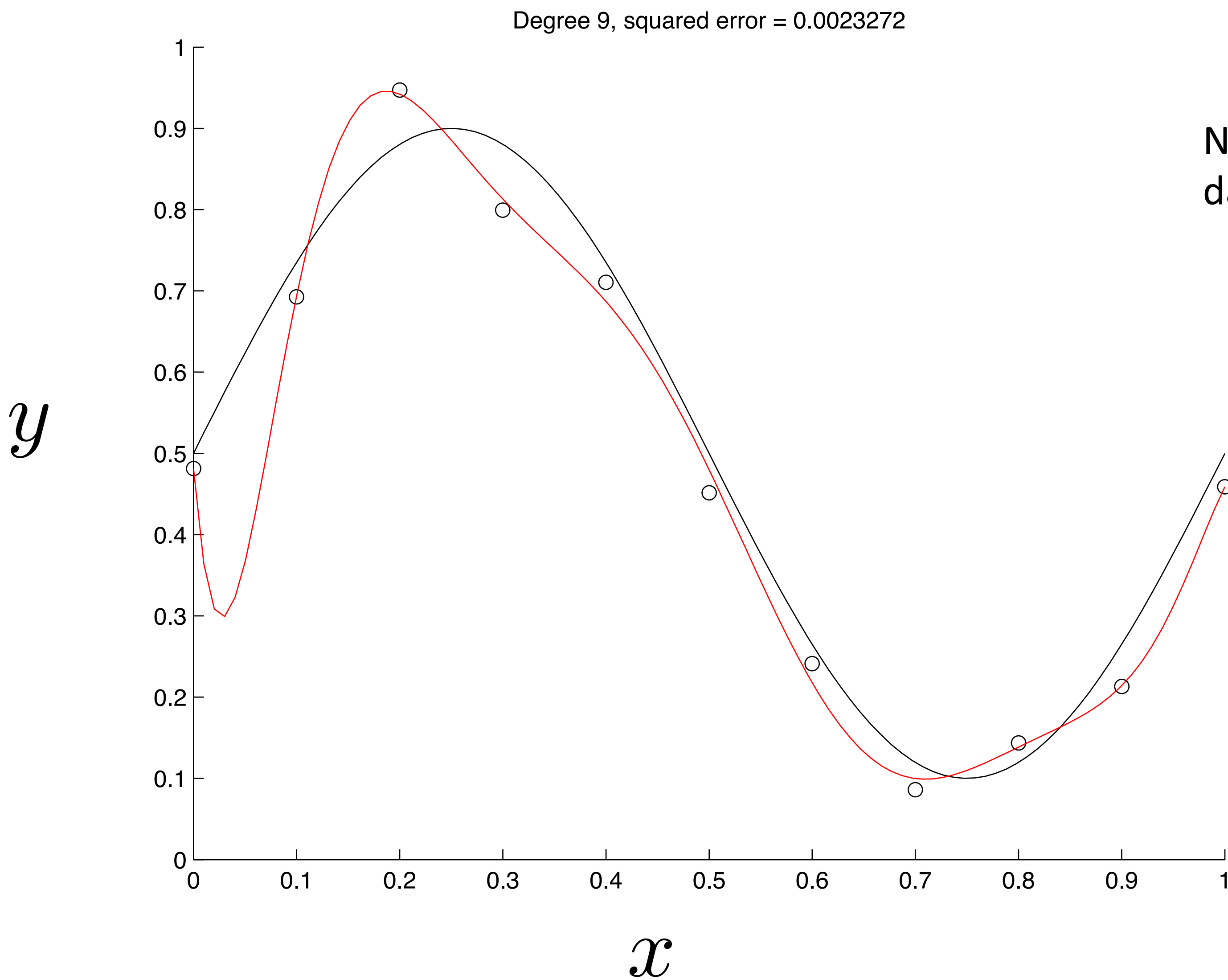
N = 11 datapoints
dashed = true fxn

Cubic



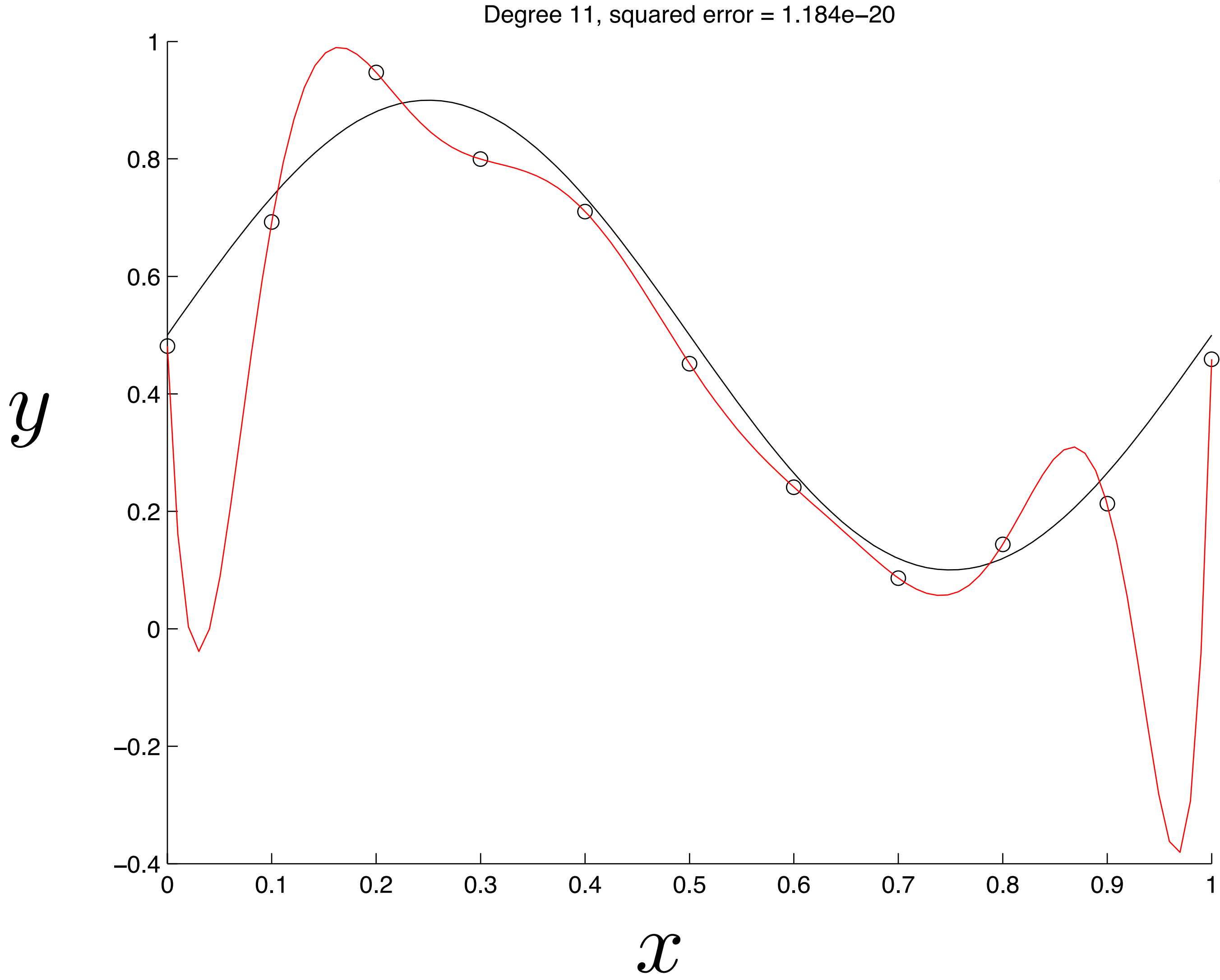
N = 11 datapoints
dashed = true fxn

M=9



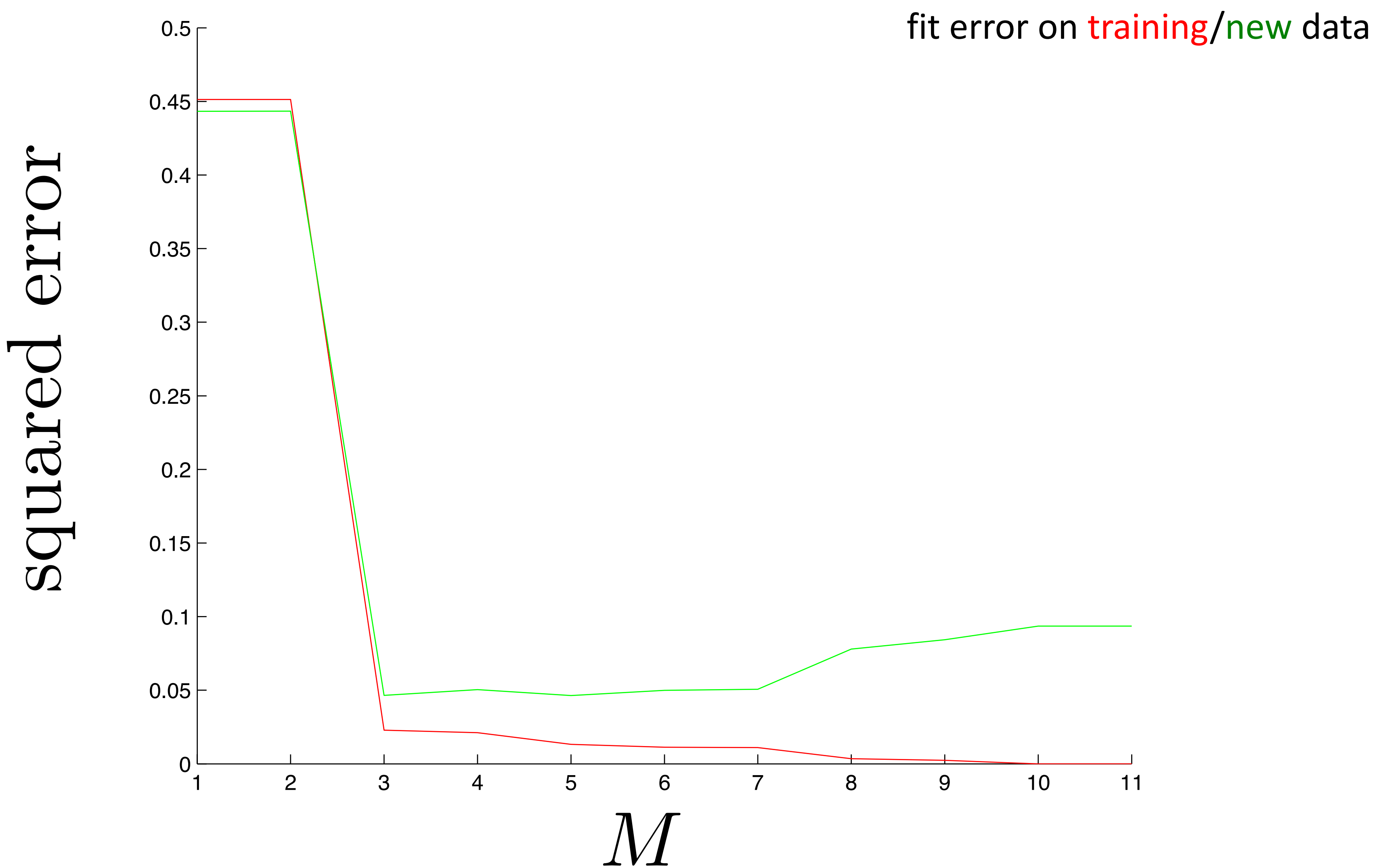
N = 11 datapoints
dashed = true fxn

M = 11



N = 11 datapoints
dashed = true fxn

Sum-of-squares error



Predictability

- Error on fitting the specific training data keeps decreasing with model complexity (M).

- Error of fit to previously un-fit/unseen data improves but then worsens with increasing M.

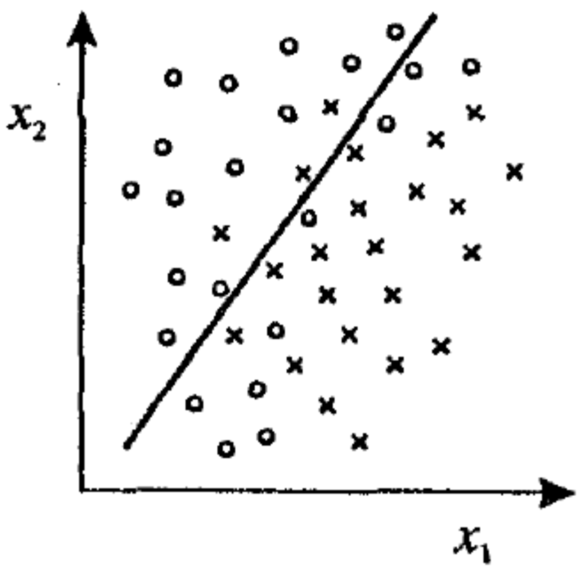
- Model is *overfitting* to foibles of training data (noise) after M = 3.

- Model becomes both *more complex* and *less predictive* beyond M = 3 features.

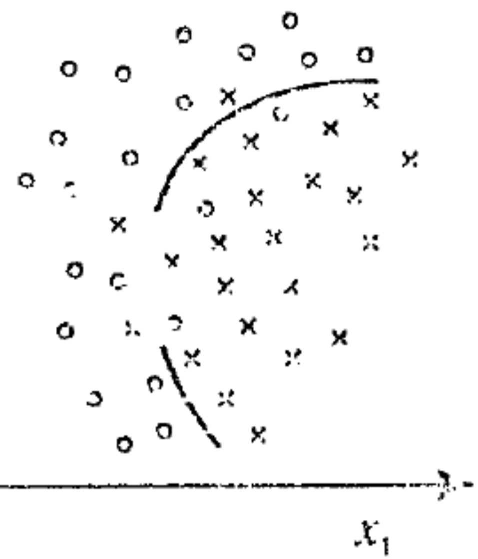
- Key technique: cross-validation. Test model on previously unseen data. Hold-out dataset or jack-knife/leave-one-out approaches.

  (There are other ways to improve predictability by reducing complexity, e.g. by directly constraining the complexity of the model: “regularization”)

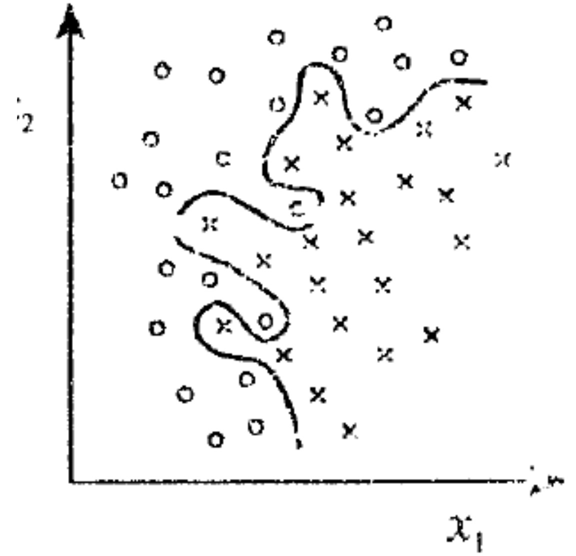
Back to categorization example



simplest

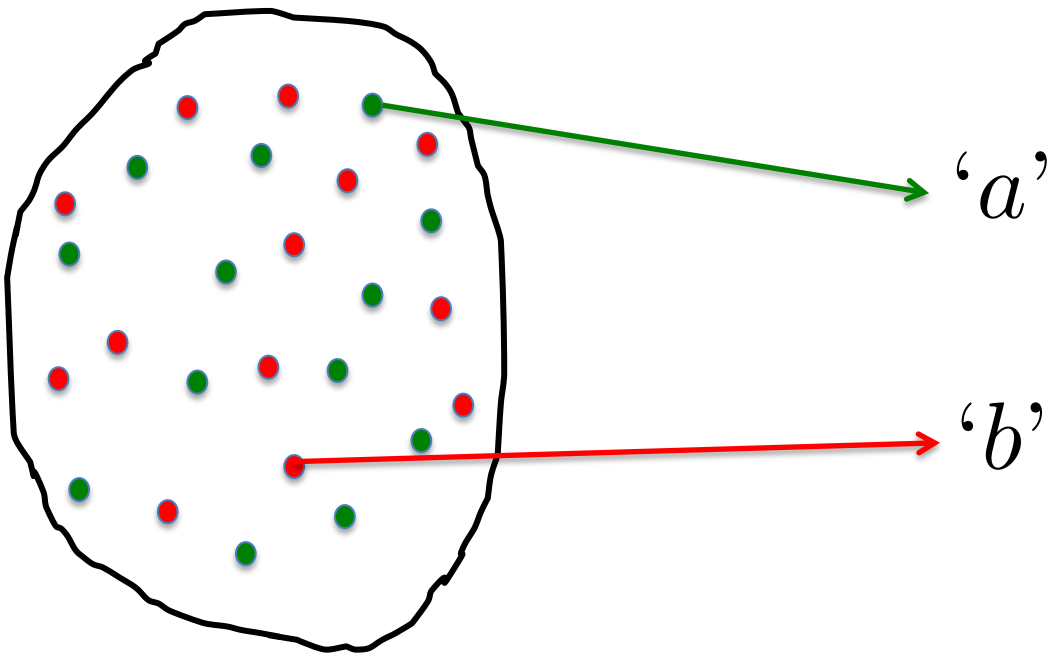


intermediate

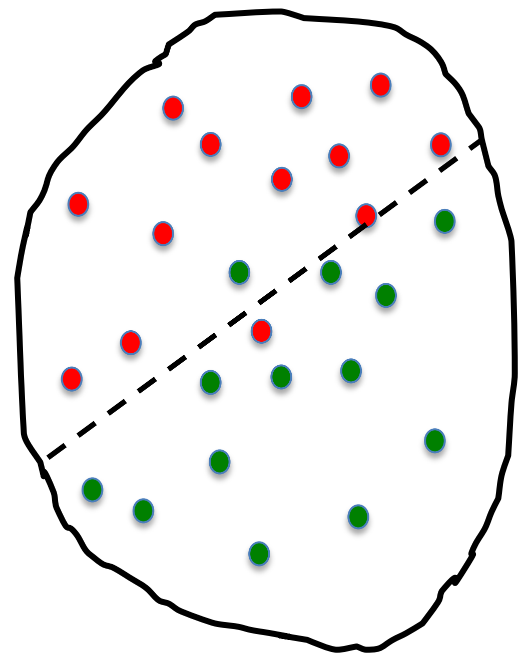


most flexible/complex
exhibits overfitting

Better features: admit simpler model



poor choice of features



better choice of features

(In regression example, data were generated from a sine wave.
Using sines instead of polynomials would have produced an excellent 2-parameter fit.)

Summary: what is modeling?

- A good model can describe the data in a relatively simple/low-complexity/compact way (but not too low! Einstein: as simple as possible, but no simpler) and has good prediction performance.

- Extracting “features” of data as a way to model it.

- To determine predictability, important to cross-validate models/fits.