A.I. in health informatics **lecture 6** pattern classification (and regression)

kevin small &
byron wallace

today

- feature-spaces
- regression
  - least-squares
- classification
  - decision trees

what's in a feature-space?

- machine learning algorithms operate over vectors
- a *feature-space* is the vector-space used to represent things for ml algorithms
- suppose we want to predict housing prices; what attributes would you use and how would you encode them?

what's in a feature-space?

- feature-engineering isn't considered 'sexy'
  - but in practice it is probably the most important contributor to performance

- ***representation is everything***!

what's in a feature-space?

- want to classify people as 'diabetic' or not
- have the following informatic
  - their name
  - their favorite color
  - the month they were born

what's in a feature-space?

suppose we are classifying objects

$$\{x_0\ x_1\ \ldots\ x_n\}$$

a feature mapping **F** maps from the input into its vector representation (in the feature-space)

text encoding

- many applications in health informatics involve text
- we will review ‘bag-of-words’ style text encoding (or, how to vectorize text)

text encoding by example

suppose we want to encode the following sentences:

S_1= “Boston drivers are frequently aggressive”
S_2= “The Boston Red Sox frequently hit line drives”

text encoding by example

1. remove stop-words

S_1= “Boston drivers are frequently aggressive”
S_2= “The Boston Red Sox frequently hit line drives”

text encoding by example

2. lower-case

S_1= "*b*oston drivers are frequently aggressive"
S_2= "The *b*oston *r*ed *s*ox frequently hit line drives"

text encoding by example

3. stem

S_1= "*b*oston drive*s* a*re* frequentl*y* aggressive"
S_2= "T*he* *b*oston *r*ed *s*ox frequentl*y* hit line drive*s*"

text encoding: indicator vectors

S_1= "*boston* drivers are frequently aggressive"
S_2= "The *boston red sox* frequently hit line drives"

V = [hit, red, sox, line, boston, frequent, drive,
aggressive]

text encoding: indicator vectors

	hit	red	sox	line	boston	frequent	drive	aggressive
S_1 =	0	0	0	0	1	1	1	1
S_2 =	1	1	1	1	1	1	1	0

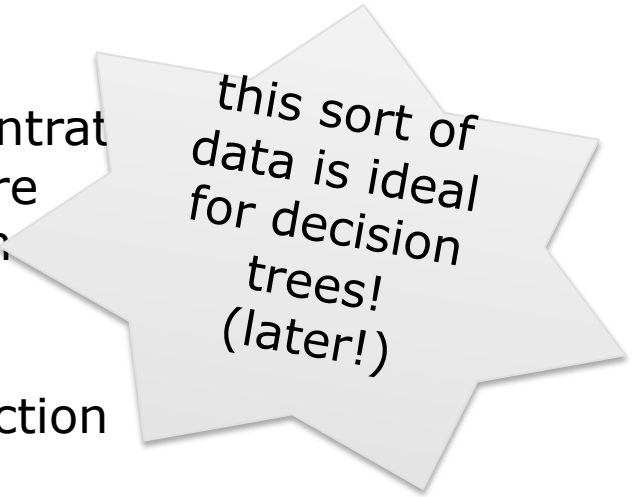
a new sentence, S_3, comes along it reads: "I hate the red sox". to which sentence is it most similar?

S_3 =	0	1	1	0	0	0	0	0

other sorts of spaces

- text tends to be *high-dimensional* and *sparse*
- pima dataset

0 # of times pregnant
1 plasma glucose concentrat
2 diastolic blood pressure
3 triceps skin fold thickn
4 2-Hour serum insulin
5 body mass index
6 diabetes pedigree function
7 age
label -1 or 1



other sorts of spaces

feature	person P
# of times pregnant	3
plasma glucose	19
concentration	4.4
diastolic blood pressure	5
triceps skin fold thickness	34
2-hour serum insulin	2
BMI	32
diabetes pedigree function	2
age	24

F(**P**) = [3, 19, 4.4, 5, 34, 2, 32, 2, 24]

normalization, etc

F(**P**) = [3, ***19***, 4.4, 5, ***34***, 2, ***32***, 2, ***24***]

- ***some*** features are getting undue weight
- solution: normalize each column (eg., divide all entries in each column by the max entry in that column)

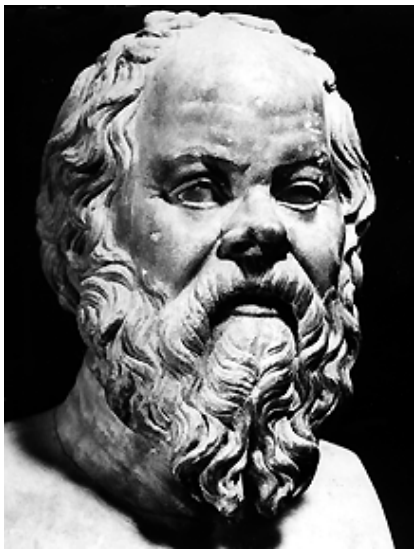
standardizing

- in addition to column normalization, we sometimes want to row-standardize, ie., enforce a zero mean unit variance on each instance **x**

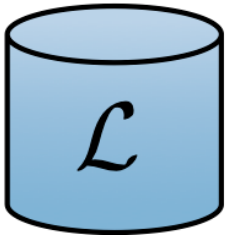
$$x'_i = \frac{x_i - \overline{x}_i}{sd(x)}$$

not necessary in general, but required for some algorithms, eg., PCA

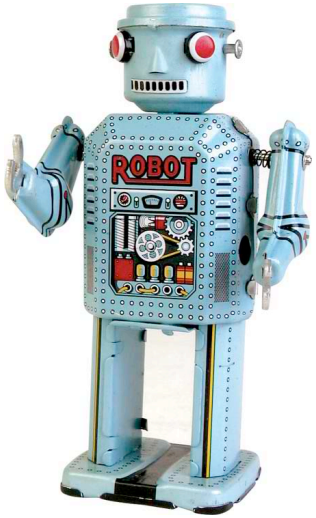
supervised learning



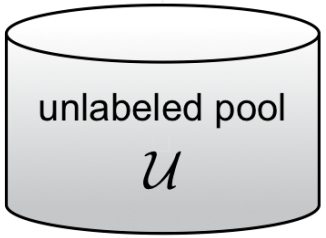
(human) expert / labeler



labels subset *L*



classifier *C* is induced over *L*



use *C* to classify unlabeled examples

generalized supervised learning

$$\arg\min_{\theta} \underbrace{\left(h(x,\theta) - y\right)^2}_{\text{minimize error}} + \underbrace{\lambda R(h,\theta)}_{\text{penalize complexity}}$$

supervised learning

- two main types of learning tasks:
  - regression
  - classification
- today start with regression, then do classification

regression

- the regression task:

  given a **training set** $\{(\mathbf{x}_0, y_0), (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$ where y_is are scalars (in **R**), induce a model **h** such that given a new instance $\mathbf{x}_{m+1}$, $\mathbf{h}(y_{m+1})$ is minimized

- eg., predict the price of a house given its attributes; predict someone's BMI given some of their physical characteristics…

regression

- many possible ways to do this
- assume that we're interested in the linear case $\mathbf{w} \bullet \mathbf{x}_i = y_i$
  - doesn't always make sense (y could be, e.g., polynomial in **w**)

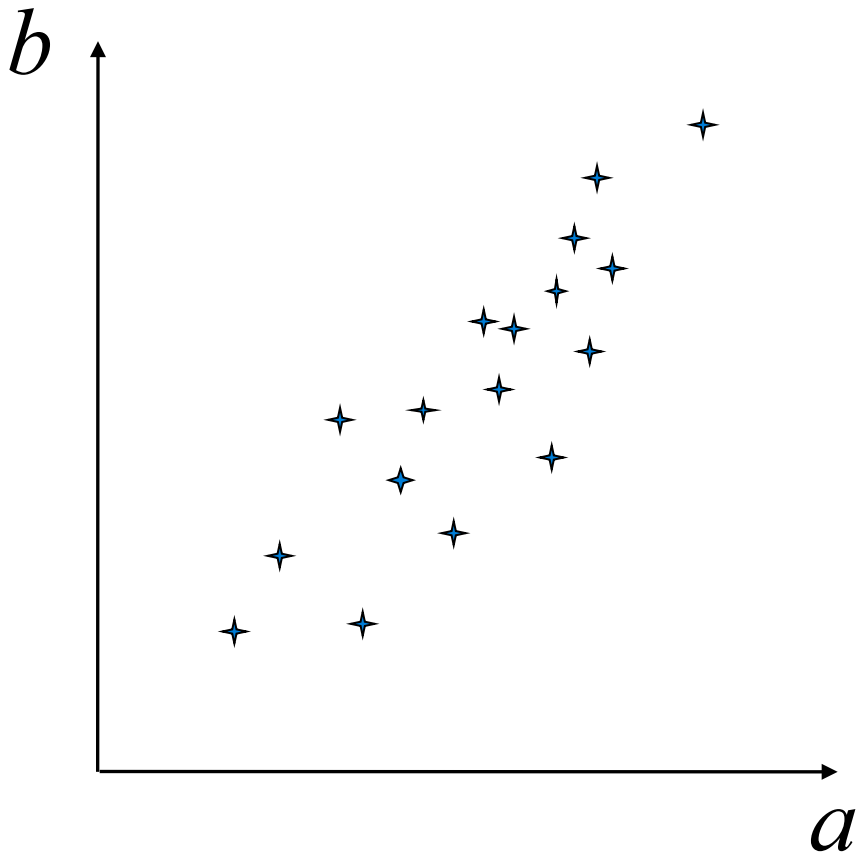
very brief linear algebra review

- vector $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]$ $\qquad \mathbf{x}^t = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}$
- dot product $\mathbf{a} \bullet \mathbf{b} = \sum_i \mathbf{a}_i \cdot \mathbf{b}_i$
- m x n matrix $\mathbf{X} = \begin{bmatrix} x_{11}\ x_{12}\ \ldots\ x_{1n} \\ x_{21}\ x_{22}\ \ldots\ x_{2n} \\ \ldots \\ x_{m1}\ x_{m2}\ldots\ x_{mn} \end{bmatrix}$

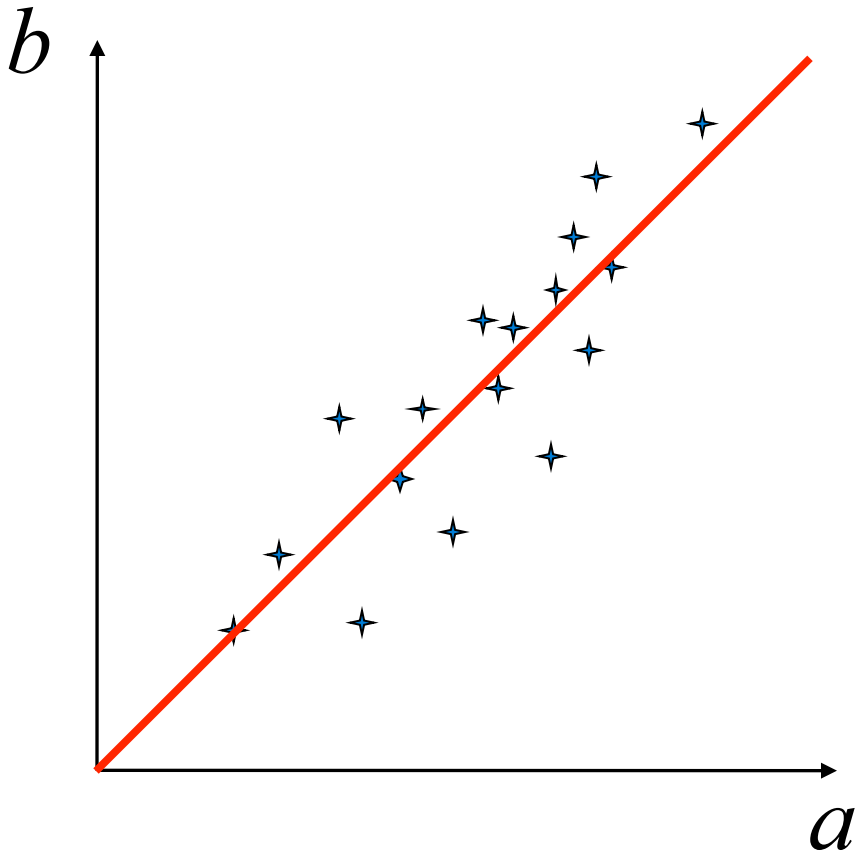
acknowledgement: many of the slides from here on (those regarding least-squares linear regression) are borrowed heavily from

graphics.stanford.edu/~jplewis/lscourse/ols_slides.ppt

one-dimensional regression



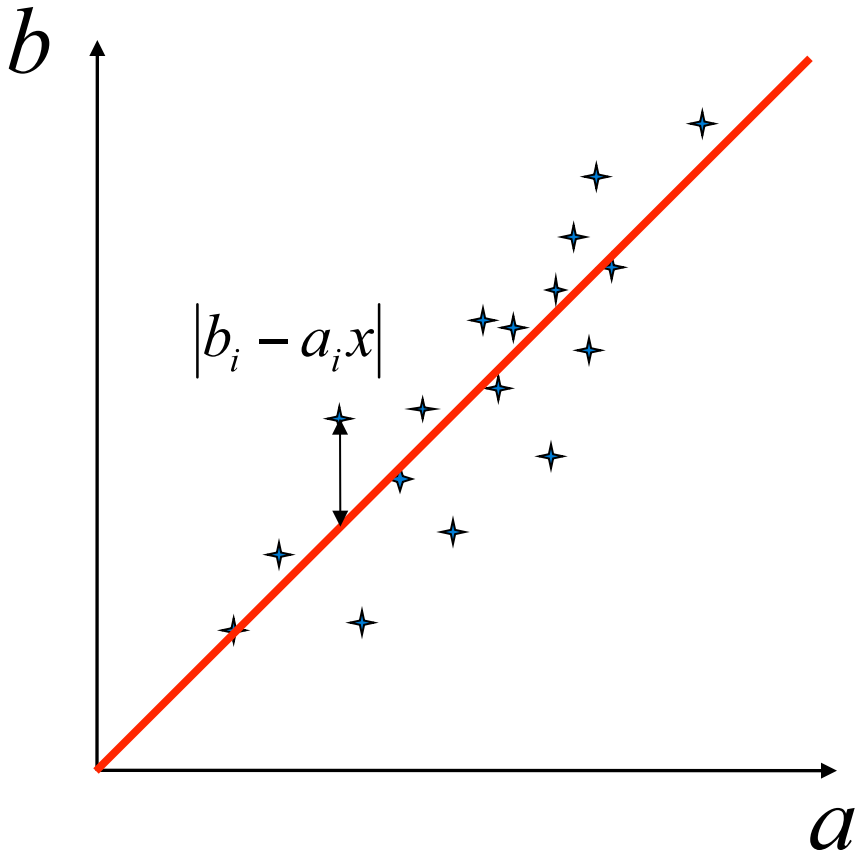
one-dimensional regression



aim: find a line that represent the ”best” linear relationship:

$$b = ax$$

one-dimensional regression

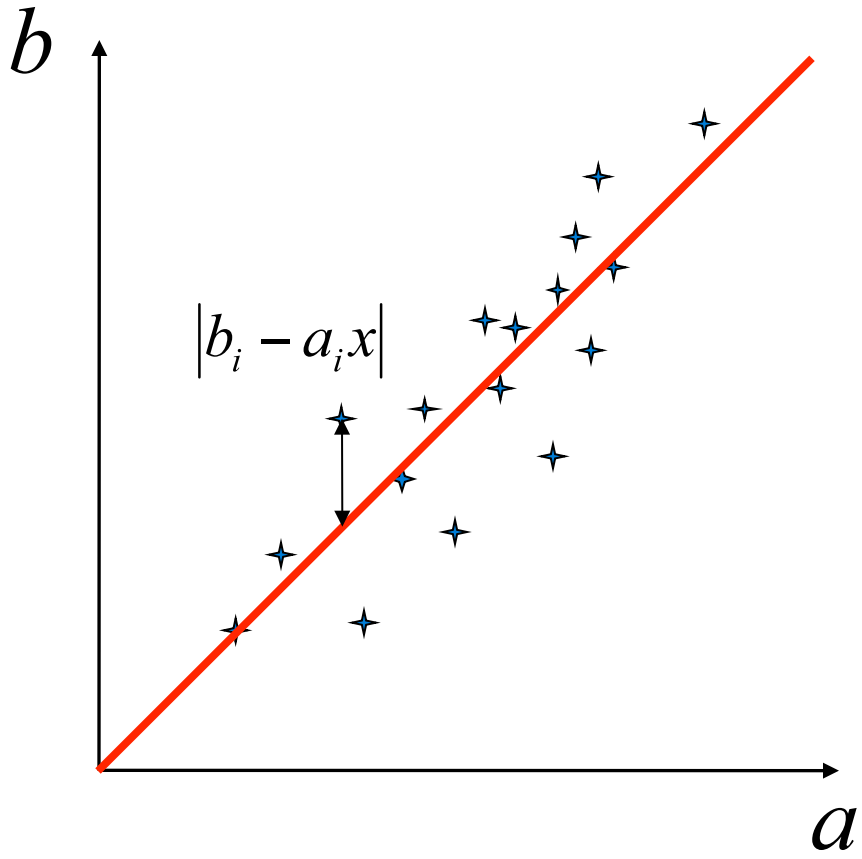


problem the data does not go through a line

the error is:

$$e_i = b_i - a_i x$$

one-dimensional regression



problem the data does not go through a line

$$e_i = b_i - a_i x$$

objective find the line that minimizes the sum:

$$\sum_i (b_i - a_i x)^2$$

… in terms of our general aim

$$\arg\min_{\theta} \underbrace{\left(h(x,\theta) - y\right)^2}_{\text{minimize error}} + \underbrace{\lambda R(h,\theta)}_{\text{penalize complexity}}$$

in the case of (one-dimensional) LS linear regression:

$$\arg\min_{x} \sum_{i} (b_i - a_i x)^2$$

(just sort of ignoring model complexity)

re-expressed in matrix notation

$$\sum_i (b_i - a_i x)^2$$

$$= (\boldsymbol{b} - x\boldsymbol{a})^T (\boldsymbol{b} - x\boldsymbol{a})$$

$$= \underbrace{\left\| \boldsymbol{b} - x\boldsymbol{a} \right\|^2}$$

this is what we want to minimize!

more generally (multidimensional linear regression)

model with *m* parameters

$$b = a_1 x_1 + \ldots + a_m x_m = \sum_j a_j x_j$$

and *n* measurements (examples)

$$e(\boldsymbol{x}) = \sum_{i=1}^{n} (b_i - \sum_{j=1}^{m} a_{i,j} x_j)^2$$

$$= \left\| \boldsymbol{b} - \left[\sum_{j=1}^{m} a_{i,j} x_j \right] \right\|^2$$

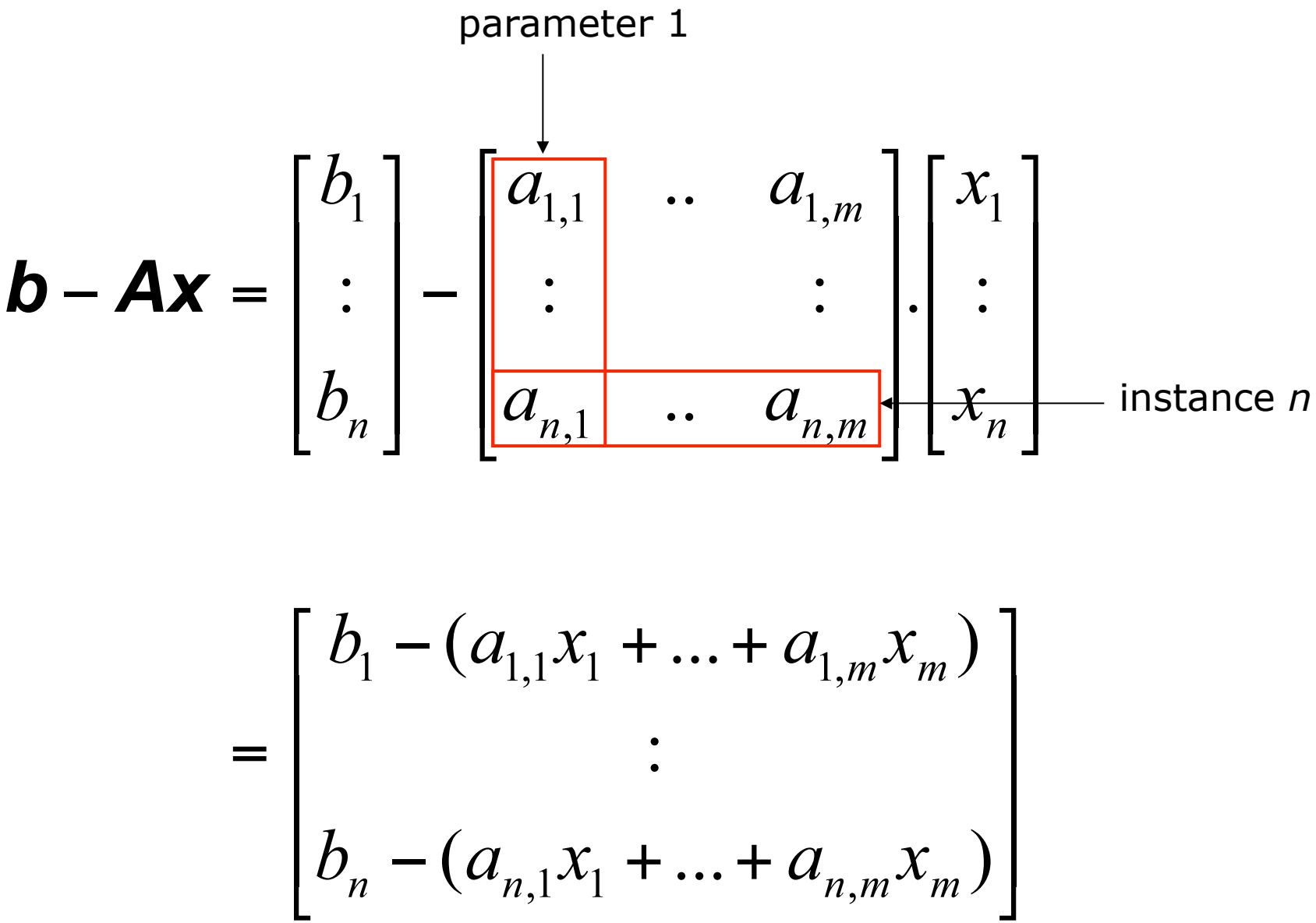
find $\boldsymbol{x}$ that minimizes this!

$$e(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2$$

minimizing the error

$$\boldsymbol{b - Ax} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & .. & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & .. & a_{n,m} \end{bmatrix} . \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

$$= \begin{bmatrix} b_1 - (a_{1,1}x_1 + \ldots + a_{1,m}x_m) \\ \vdots \\ b_n - (a_{n,1}x_1 + \ldots + a_{n,m}x_m) \end{bmatrix}$$


parameter 1
$\boldsymbol{b - Ax} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & .. & a_{1,m} \\ \vdots & & \vdots \\ a_{n,1} & .. & a_{n,m} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$
instance *n*
$= \begin{bmatrix} b_1 - (a_{1,1}x_1 + \ldots + a_{1,m}x_m) \\ \vdots \\ b_n - (a_{n,1}x_1 + \ldots + a_{n,m}x_m) \end{bmatrix}$

solving for **x**

$$e = (\boldsymbol{b} - x\boldsymbol{A})^T(\boldsymbol{b} - x\boldsymbol{A})$$

$$= \boldsymbol{b}^T\boldsymbol{b} - 2\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}$$

solving for **x**

as usual, we take the derivative, set it to 0 and solve for the variable of interest (**x**)

$$\frac{\partial e}{\partial \mathbf{x}} = -2A^t b + 2A^t Ax$$

$$0 = -2A^t b + 2A^t Ax$$

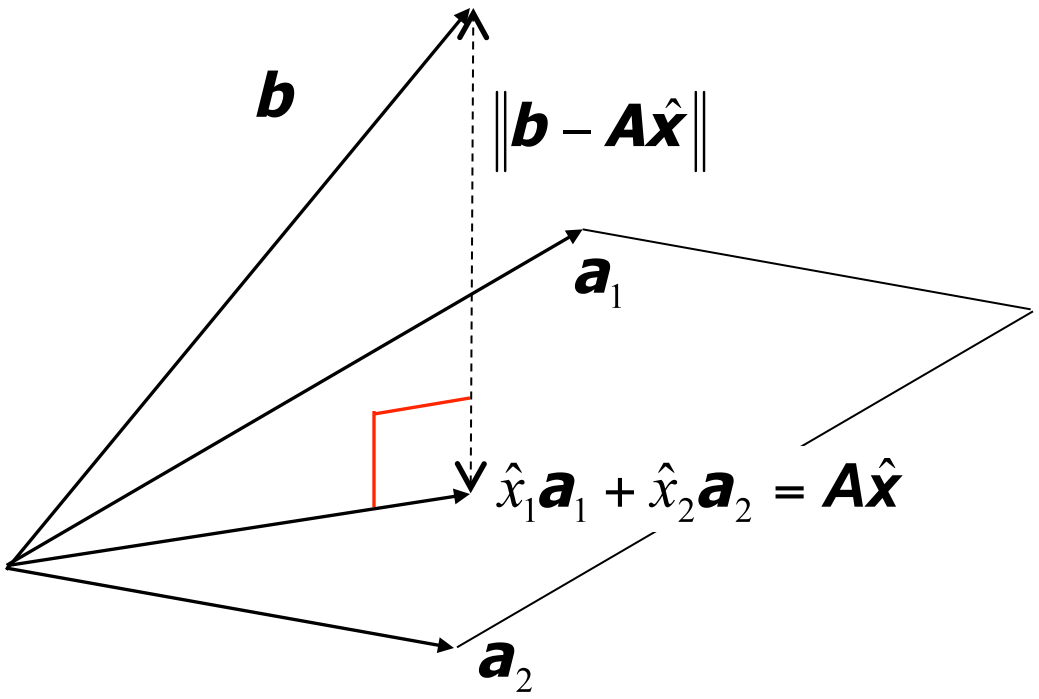
$$\rightarrow A^t Ax = A^t b$$

$$\rightarrow \hat{x} = (A^t A)^{-1} A^t b$$

the geometric interpretation

- $\boldsymbol{A\hat{x}}$ is the orthogonal projection of $\boldsymbol{b}$ onto *range*($\boldsymbol{A}$)

$$\Leftrightarrow \boldsymbol{A}^T\left(\boldsymbol{b}-\boldsymbol{A}\hat{\boldsymbol{x}}\right)=\boldsymbol{0} \Leftrightarrow \boldsymbol{A}^T\boldsymbol{A}\hat{\boldsymbol{x}}=\boldsymbol{A}^T\boldsymbol{b}$$



awesome!
wait, what were we doing again?

given a new instance, **a** our prediction is:

$$\hat{\boldsymbol{x}} \cdot \boldsymbol{a}$$

what about penalizing for complexity?

- here we sort of ignored model complexity
- how might we penalize for this?
- ***Lasso*** constrain the norm of **x,** our parameter vector, to be <= *r*

that’s it for regression

- often we want to *classify* things into categories, as opposed to predicting a continuous value

- this is known as *classification* (next class!)