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 information:
  – 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:

\[ S_1 = \text{“Boston drivers are frequently aggressive”} \]
\[ S_2 = \text{“The Boston Red Sox frequently hit line drives”} \]
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

$S_1 =$ “boston drivers are frequently aggressive”

$S_2 =$ “The boston red sox frequently hit line drives”
text encoding by example

3. stem

$S_1 =$ “*boston drivers are frequently aggressive*”
$S_2 =$ “The *boston red sox frequently hit line drives*”
text encoding: indicator vectors

\[ S_1 = \text{“}boston\text{ drivers frequently aggressive”} \]
\[ S_2 = \text{“}The\ boston\ red\ sox\ frequently\ hit\ line\ drives\” \]

\[ V = [\text{hit, red, sox, line, boston, frequent, drive, aggressive}] \]
text encoding: indicator vectors

<table>
<thead>
<tr>
<th></th>
<th>hit</th>
<th>red</th>
<th>sox</th>
<th>line</th>
<th>boston</th>
<th>frequent</th>
<th>drive</th>
<th>aggressive</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$S_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A new sentence, $S_3$, comes along it reads: “I hate the red sox”. To which sentence is it most similar?

$S_3 = [0, 1, 1, 0, 0, 0, 0, 0, 0]$
other sorts of spaces

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

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

**label** -1 or 1

this sort of data is ideal for decision trees! (later!)
other sorts of spaces

\[
\begin{array}{|c|c|}
\hline
\text{feature} & \text{person P} \\
\hline
\text{of times pregnant} & 3 \\
\text{plasma glucose} & 19 \\
\text{concentration} & 4.4 \\
\text{diastolic blood pressure} & 5 \\
\text{triceps skin fold thickness} & 34 \\
\text{2-hour serum insulin} & 2 \\
\text{BMI} & 32 \\
\text{diabetes pedigree function} & 2 \\
\text{age} & 24 \\
\hline
\end{array}
\]

\[
\mathbf{F}(P) = [3, 19, 4.4, 5, 34, 2, 32, 2, 24]
\]
normalization, etc

\[ F(\mathbf{P}) = [3, 19, 4.4, 5, 34, 2, 32, 2, 24] \]

- **some** features are getting undue weight

- solution: normalize each column (eg., divide all entries in each column by the max entry in that column)
in addition to column normalization, we sometimes want to row-standardize, i.e., enforce a zero mean unit variance on each instance $\mathbf{x}$

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

not necessary in general, but required for some algorithms, e.g., PCA
supervised learning

(human) expert / labeler

labels subset $L$

classifier $C$ is induced over $L$

unlabeled pool $U$

use $C$ to classify unlabeled examples
generalized supervised learning

$$\arg\min_{\theta} (h(x, \theta) - y)^2 + \lambda R(h, \theta)$$

- minimize error
- 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** \( \{(x_0, y_0), (x_1, y_1) \ldots (x_m, y_m)\} \)
where \( y_i \)s are scalars (in \( \mathbb{R} \)), induce a model \( h \) such
that given a new instance \( x_{m+1}, h(y_{m+1}) \) 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 $w \cdot x_i = y_i$
  – doesn’t always make sense (y could be, e.g., polynomial in $w$)
very brief linear algebra review

- vector $\mathbf{x} = [x_1 \ x_2 \ \ldots \ x_n]$  \hspace{1cm} $\mathbf{x}^t = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

- dot product $\mathbf{a} \cdot \mathbf{b} = \sum_i a_i \cdot b_i$

- m x n matrix $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{bmatrix}$
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

\[
\arg\min_{\theta} (h(x, \theta) - y)^2 + \lambda R(h, \theta)
\]

minimize error  \hspace{1cm} \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

\[
\sum_i (b_i - a_i x)^2 = (b - x a)^T (b - x a) = \| b - x a \|^2
\]

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)

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

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

find $x$ that minimizes this!
minimizing the error

\[ b - Ax = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \]

\[ = \begin{bmatrix} b_1 - (a_{1,1}x_1 + \cdots + a_{1,m}x_m) \\ \vdots \\ b_n - (a_{n,1}x_1 + \cdots + a_{n,m}x_m) \end{bmatrix} \]
\[ b - A x = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \]

\[ = \begin{bmatrix} b_1 - (a_{1,1}x_1 + \cdots + a_{1,m}x_m) \\ \vdots \\ b_n - (a_{n,1}x_1 + \cdots + a_{n,m}x_m) \end{bmatrix} \]
solving for $\mathbf{x}$

\[ e = (\mathbf{b} - x\mathbf{A})^T (\mathbf{b} - x\mathbf{A}) \]

\[ = \mathbf{b}^T \mathbf{b} - 2\mathbf{b}^T \mathbf{A} \mathbf{x} + x^T \mathbf{A}^T \mathbf{A} \mathbf{x} \]
solving for $\mathbf{x}$

as usual, we take the derivative, set it to 0 and solve for the variable of interest ($\mathbf{x}$)

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

$$0 = -2\mathbf{A}^t \mathbf{b} + 2\mathbf{A}^t \mathbf{A} \mathbf{x}$$

$$\rightarrow \mathbf{A}^t \mathbf{A} \mathbf{x} = \mathbf{A}^t \mathbf{b}$$

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

- \( A\hat{x} \) is the orthogonal projection of \( b \) onto \( \text{range}(A) \)
  \[
  \iff A^T(b - A\hat{x}) = 0 \iff A^T A\hat{x} = A^T b
  \]
awesome!
wait, what were we doing again?

given a new instance, our prediction is:

\[ \hat{x} \cdot a \]
what about penalizing for complexity?

• here we sort of ignored model complexity

• how might we penalize for this?

• **Lasso** constrain the norm of $\mathbf{x}$, our parameter vector, to be $\leq r$
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!)