Lec 2

Tuesday, September 3, 2019 10:53

Supervised learning:

Given examples $X_1, Y_1, \cdots, X_n, Y_n$

Want to learn a prediction rule f

s.t. for any new unseen example

$$Y \approx f(X) = \hat{Y}$$

$$X \to \boxed{f} \to \hat{Y} \approx Y$$

The output Y takes values in G

We classify the cases as:

- $G = \{a, b\}$ binary classification

  where a, b can be anything

  e.g:
  - $G = \{0, 1\}$
  - $G = \{\text{red}, \text{blue}\}$
  - $G = \{\text{high risk}, \text{low risk}\}$
- $G = \{a, b, c, \cdots\}$ multi-class classification
- $G = \mathbb{R}$ regression

Our first classification algo: KNN

Given a new query point x:

- Find the k "closest" x-values in

Given a new query point x
- Find the k "closest" x-values in the examples $x_1, \dots, x_n$
- Note their indices $i_1, \dots, i_k \in [n] = \{1, \dots, n\}$
- classify the example as the most common label in the corresponding set of training labels

$$\hat{Y} = \text{mode}(\{Y_{i_1}, \dots, Y_{i_k}\})$$

What does "closest" mean?
Need a distance measure.

Suppose X is a vector of features

$$X_i = \begin{pmatrix} X_{i1} \\ \vdots \\ X_{ip} \end{pmatrix} \in \mathbb{R}^p$$

Can use Euclidean distance

$$\|x - x'\|_2 = \sqrt{\textstyle\sum_j (x_j - x_j')^2}$$

or can use any other vector distance

e.g. $\|x - x'\|_\infty = \max_j |x_j - x_j'|$

Constructing feature vectors

1 If using these vector distances, mean

1. If using these vector distances, may want to first standardize the data:

$$x_{ij} \leftarrow (x_{ij} - \hat{\mu}_j)/\hat{\sigma}_j$$

mean & std dev of $x_{1j}, \ldots, x_{nj}$

2. If some of the features are categorical

e.g. Which of {Yale, Columbia, Cornell} did you attend

often we encode this using

<u>one-hot</u> encoding:

use <u>three</u> binary variables an encode

Yale as $\begin{pmatrix}1\\0\\0\end{pmatrix}$ Columbia $\begin{pmatrix}0\\1\\0\end{pmatrix}$ Cornell $\begin{pmatrix}0\\0\\1\end{pmatrix}$

3. Later in class, we'll also talk about more abstract ways to measure similarity.

E.g. X = Content of an email in words

Y = whether spam

Out-of-sample evaluation

kNN tries to make $\hat{Y}$ equal Y

kNN tries to make $\hat{y}_i$ equal Y

To evaluate how well it — or any other supervised learning algo — does is to ask how often is that true.

If X, Y are r.v.s representing new random examples drawn from the population of examples, then we want low risk

$$R(f) = \mathbb{E}[\, l(Y, f(X))\,]$$

where l is a <u>loss fn</u>

For now: $l(y, \hat{y}) = \mathbb{I}[y \neq \hat{y}] = \begin{cases} 1 & y \neq \hat{y} \\ 0 & y = \hat{y} \end{cases}$

To estimate $R(f)$ we can take a test set of examples $X_1^{test}, Y_1^{test}, \ldots, X_{n_{test}}^{test}, Y_{n_{test}}^{test}$ drawn at random from the population of examples & compute an empirical avg

$$\hat{R}_{n_{test}}(f) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} l(Y_i^{test}, f(X_i^{test}))$$

This is a consistent & unbiased

estimate of $R(f)$.

Often, we get the test set by splitting off from the training data.

Bayes classifier & Bayes (Error) Rate

So kNN is just one classifier.
But: What is the **BEST** classifier?
The one that minimizes $R(f)$

$$R(f) = \mathbb{E}[\ell(Y, f(X))]$$
$$= \mathbb{E}[\mathbb{I}[Y \neq f(X)]]$$
$$= \mathbb{E}[1 - \mathbb{I}[Y = f(X)]]$$
$$= \mathbb{E}[\mathbb{E}[1 - \mathbb{I}[Y = f(X)] \mid X]] \quad \text{(total expectation)}$$
$$= \mathbb{E}\left[\sum_{y \in G} \mathbb{P}(Y = y \mid X)(1 - \mathbb{I}[y = f(X)])\right]$$
$$= 1 - \mathbb{E}\left[\sum_{y \in G} \mathbb{P}(Y = y \mid X)\, \mathbb{I}[y = f(X)]\right]$$

For each x, we can choose one $y \in G$ so that $f(x) = y$

Which y to choose to minimize $R(f)$?
Maximize the 2^{nd} term

for each x, choose $f(x)$ to maximize

$$\sum_{y \in G} \mathbb{P}(Y=y \mid X=x)\, \mathbb{I}[y=f(x)]$$

$\Rightarrow$ choose $f(x)$ to be the maximizer of $\mathbb{P}(Y=y \mid X=x)$

$$f^*(x) = \underset{y \in G}{\operatorname{argmax}}\ \mathbb{P}(Y=y \mid X=x) = \text{mode}\big(\mathbb{P}(Y \mid X=x)\big)$$

f^* is known as the Bayes classifier

$R(f^*)$ is known as the Bayes (error) rate

the best possible risk

f^* gets at the fundamental limit of the predictability of Y from X

kNN as a probabilistic classifier

Can modify the output of kNN as a probability estimate:

$$\hat{\mathbb{P}}(Y=y \mid X) = \frac{1}{k} \sum_{j=1}^{k} \mathbb{I}[Y_{i_k}=y]$$

fraction of k nearest neighbors that have fraction of ... neighbors that have label y

Our kNN prediction can be written

$$\hat{Y} = \underset{y \in G}{\arg\max}\ \hat{\mathbb{P}}(Y=y \mid X)$$

Mimics Bayes classifier w/ an estimated conditional prob

For $G = \{0,1\}$ (binary classification), this translates to

$$\hat{Y} = \begin{cases} 0 & \hat{\mathbb{P}}(Y=1 \mid X) \le \frac{1}{2} \\ 1 & \hat{\mathbb{P}}(Y=1 \mid X) > \frac{1}{2} \end{cases}$$

$$= \mathbb{I}\left[\hat{\mathbb{P}}(Y=1 \mid X) > \tfrac{1}{2}\right]$$

Can modify this and use other thresholds

$$\hat{Y} = \mathbb{I}\left[\hat{\mathbb{P}}(Y=1 \mid X) > \theta\right] \qquad \theta \in \mathbb{R}$$

What this does is change the <u>classification rates</u>.

<u>Classification rates: binary case</u>

Classification rates: binary case

Exhaustively categorize all cases of prediction:

–	$y=1, \hat{y}=1$	TP	T = true
–	$y=0, \hat{y}=1$	FP	F = false
–	$y=0, \hat{y}=0$	TN	P = positive
–	$y=1, \hat{y}=0$	FN	N = negative

$$\#TP = \sum_{i=1}^{n_{test}} \mathbb{I}[\,y_i^{test}=1,\ f(x_i^{test}) = 1\,]$$

$\#FP = \ \dots$

$\dots$

Confusion matrix:

$\hat{y}$ \ y	1	0
1	#TP	#FP
0	#FN	#TN

Sometimes raw numbers

Sometimes divide everything by n_{test}

Rates:

True positive rate $= TPR = \frac{\#TP}{\#P} = \frac{\#TP}{\#TP + \#FN}$

aka Recall

$TNR = \frac{\#TN}{\;} = \frac{\#TN}{\;}$

$$TNR = \frac{\# TN}{\# N} = \frac{\# TN}{\# TN + \# FP}$$

$$FPR = 1 - TNR$$
$$= \frac{\# FP}{\# TN + \# FP}$$

$$FNR = 1 - TPR = \frac{\# FN}{\# TP + \# FN}$$

Accuracy $Acc = 1 - (\text{Avg 0-1 loss})$
$$= \frac{\# TP + \# TN}{\# TP + \# TN + \# FP + \# FN}$$