Announcements

Graded HW2 will be available tomorrow after 9:30 AM from Anna (RM 203)

4/24/08

Today: Neighborhood Methods

- Diff between model-based and neighborhood approaches
- Theoretical performance bounds
- Different variations on classifier/regression themes
- Practical issues

Reminder of previous approaches:

1) Assume a probability distribution, estimate parameters, apply decision theory
2) Assume a decision function, form a estimate parameters

Both cases are model based
In approaches so far, we use all the data to get the decision function.

Consider the idea of having the data locally determine the decision function.
Model-based:
- Use all training data to find separating line
- Throw away data
- Classify based on line

Classify ?'s as X or O
Classify ?’s as X or O

Neighborhood-based
- Classify based on neighborhood
  (local data)
- Don't throw away any training data, ignore non-local data
Model - Band
- \( \hat{y} = f(x) = \beta^x \) choose model
- use all the data to find \( \beta \)
- Throw away data
- Classify with my function
Neighborhood based:

- Find local region
- Use (weighted) average of y_i in that region
Theoretical Classifier Performance Bounds

We know the best we can possibly do is Bayes Risk. For min error (no cost problems) call this $P^*$ (know true distri.)

$$P^*(e|x) = 1 - P(w_m|x)$$

max prob class

$1-NN$ with $C$ classes

$x = \text{test sample}$

$x_n' = \text{nearest neighbor out of n samples}$

$$P(e|x, x_n') = \left(1 - \frac{1}{C}, p(w_i|x) p(w_i|x_n') \right)$$

$$P_e(x) = \int_p(e|x, x') p(x'|x) dx'$$

$P_n(e|x, x_n') \rightarrow P(e|x, x')$

$p_n(x'|x) \rightarrow \delta(x'-x)$

$$\lim_{n \rightarrow \infty} P_{e_n} = 1 - \frac{1}{C_0} \int_{C_0} P^2(w|x) p(x) dx$$
how does $p$ compare to $p^*$?

Details in DHS

use clever bounds on $p(w|x)$

$$\Rightarrow p^* \leq p \leq p^* (2 - \frac{5}{c-1} p^*)$$

for $c = 2$

$$p^* \leq p \leq 2p^* (1 - p^*) \leq 2p^*$$

Worst you can do with NO ASSUMPTIONS is $2p^*$

\[ \frac{1}{2p^*} < 1 - NN \]

Increasing $K$
To get $P^*$ with K-NN, you need

$$K \to \infty, \ N \to \infty, \ \frac{K}{N} \to 0 \ (e.g. \ k = \sqrt{N})$$

Great in theory: can get a perfect answer but it will take forever to get it.

Important to consider practical trade-offs.

We'll have 2 problems:

- How do you choose $K$?
- How to deal with the computation?
One more variation...

In these examples, we treated all elements in the neighborhood equally, but maybe closer points should be treated more.

\[ \Rightarrow \text{ idea of weights or kernel functions} \]

<table>
<thead>
<tr>
<th>Classification</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>argmax [ \sum_{i=1}^{n} I(x_i \in G, x_i \in S(x, h)) \hat{f}(x) ]</td>
<td>Uniform [ \hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} 1(x \in S(x, h), y_{i}) ]</td>
</tr>
<tr>
<td>Indicator (majority vote)</td>
<td>neigborhood</td>
</tr>
</tbody>
</table>

argmax \[ \sum_{i=1}^{n} \psi(\frac{x-x_i}{h}) I(x_i \in G) \] \[ \hat{f}(x) = \frac{1}{n} \sum_{i=0}^{n} \psi(\frac{x-x_i}{h}) y_i \] 

\[ \psi(\cdot) : \text{kernel function} \]

\[ h : \text{"width"} \]
FIGURE 2.3. The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1), and then predicted by 1-nearest-neighbor classification.

FIGURE 2.2. The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable (GREEN = 0, RED = 1) and fit by 15-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the 15-nearest neighbors.
Figure from book

Low model complexity  high

prediction error

low variance

high bias

high variance

low bias

If k is too big, then your decision is biased to the majority class.
KNN is good (near boundary)
Small $k$ is good
The problem of "boundaries"

Two types of boundaries:
- between classes
- at the edge of the space

bias at the boundaries, since samples don't surround it

In high dimensions, many points are at the boundaries.
Practical Issues

Finding \( K \) (or \( h \) in kernel approach)

- If I optimize performance on the training data, what is the best \( K \)? \( K=1 \)

- So, need to try different \( K \)'s by evaluating on held out data

| training | held out | test |

Dealing with computation

- Tree search-based pruning based on partial score
- Data editing/pruning/condensing
  - merge points that close
  - remove points in internal regions (i.e., not at the boundary)
- Dimension reduction
Trade-offs

Model-based
- higher offline learning cost
- need to assume a model
- compute + space
- no guarantees since model could be wrong

Neighborhood
- higher use cost
- no assumptions necessary
- compute + space
- in theory can get close to Bayes

Same
- both have bias/variance trade-offs
- more data helps everything
- both suffer from overfitting
- benefit from model order selection (parameter tuning)
- both are affected by feature space